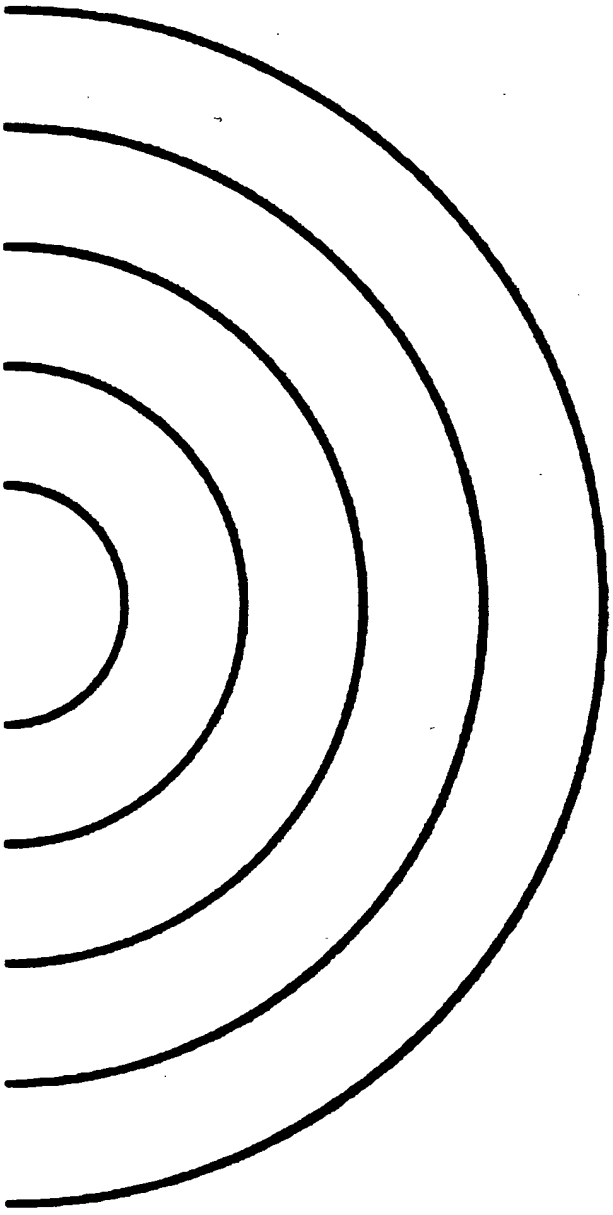
APPENDIX A

Specification Volume 1

Specification of the Bluetooth System



0355555-0120-02

Wireless connections made easy

Core

Bluetooth.

v1.0 B
December 1st 1999

BLUETOOTH DOC | 01 Dec 99 | 1.C.47/1.0 B

Bluetooth.

Specification of the Bluetooth System

Version 1.0 B

0 3 5 5 5 5 - 0 4 0 2 0 3

Bluetooth.

Revision History

The Revision History is shown in Appendix I on page 868

Contributors

The persons who contributed to this specification are listed in Appendix II on page 879.

Web Site

This specification can also be found on the Bluetooth website:
http://www.bluetooth.com

Disclaimer and copyright notice

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Copyright © 1999

Telefonaktiebolaget LM Ericsson,
International Business Machines Corporation,
Intel Corporation,
Nokia Corporation,
Toshiba Corporation .

*Third-party brands and names are the property of their respective owners.

Bluetooth.

MASTER TABLE OF CONTENTS

For the Bluetooth Profiles, *See Volume 2.*

Part A **Volume 1 (1:2)**

RADIO SPECIFICATION

Contents **[A] 17**

1 Scope [A] 18
2 Frequency Bands and Channel Arrangement [A] 19
3 Transmitter Characteristics [A] 20
4 Receiver Characteristics [A] 24
5 Appendix A [A] 28
6 Appendix B [A] 31

Part B **Volume 1 (1:2)**

BASEBAND SPECIFICATION

Contents **[B] 35**

1 General Description [B] 41
2 Physical Channel [B] 43
3 Physical Links [B] 45
4 Packets [B] 47
5 Error Correction [B] 67
6 Logical Channels [B] 77
7 Data Whitening [B] 79
8 Transmit/Receive Routines [B] 81
9 Transmit/Receive Timing [B] 87
10 Channel Control [B] 95
11 Hop Selection [B] 127
12 Bluetooth Audio [B] 139
13 Bluetooth Addressing [B] 143
14 Bluetooth Security [B] 149
15 List of Figures [B] 179
16 List of Tables [B] 183

Bluetooth.

Part C **Volume 1 (1:2)**

LINK MANAGER PROTOCOL

Contents **[C] 187**

1 General [C] 191
2 Format of LMP [C] 192
3 The Procedure Rules and PDUs [c] 193
4 Connection Establishment [C] 225
5 Summary of PDUs [C] 226
6 Test Modes [C] 237
7 Error Handling [C] 239
8 List of Figures [C] 241
9 List of Tables [C] 243

Part D **Volume 1 (1:2)**

LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

Contents **[D] 247**

1 Introduction [D] 249
2 General Operation [D] 253
3 State Machine [D] 258
4 Data Packet Format [D] 272
5 Signalling [D] 275
6 Configuration Parameter Options [D] 289
7 Service Primitives [D] 295
8 Summary [D] 313
9 References [D] 314
10 List of Figures [D] 315
11 List of Tables [D] 316
Terms and Abbreviations [D] 317
Appendix A: Configuration MSCs [D] 318
Appendix B: Implementation Guidelines [D] 321

Bluetooth.

Part E **Volume 1 (1:2)**

SERVICE DISCOVERY PROTOCOL (SDP)

Contents **[E] 325**

1 Introduction [E] 327
2 Overview [E] 330
3 Data Representation [E] 341
4 Protocol Description [E] 344
5 Service Attribute Definitions [E] 358
Appendix A– Background Information [E] 370
Appendix B – Example SDP Transactions [E] 371

Part F:1 **Volume 1 (1:2)**

RFCOMM WITH TS 07.10

Contents **[F:1] 387**

1 Introduction [F:1] 389
2 RFCOMM Service Overview [F:1] 391
3 Service Interface Description [F:1] 395
4 TS 07.10 Subset Supported by RFCOMM [F:1] 396
5 TS 07.10 Adaptations for RFCOMM [F:1] 398
6 Flow Control [F:1] 403
7 Interaction with Other Entities [F:1] 405
8 References [F:1] 408
9 Terms and Abbreviations [F:1] 409

Part F:2 **Volume 1 (1:2)**

IrDA INTEROPERABILITY

Contents **[F:2] 413**

1 Introduction [F:2] 414
2 OBEX Object and Protocol [F:2] 417
3 OBEX over RFCOMM [F:2] 421
4 OBEX over TCP/IP [F:2] 423
5 Bluetooth Application Profiles using OBEX [F:2] 425
6 References [F:2] 427
7 List of Acronyms and Abbreviations [F:2] 428

Bluetooth.

Part F:3 **Volume 1 (1:2)**

TELEPHONY CONTROL PROTOCOL SPECIFICATION

Contents **[F:3] 431**

1 General Description [F:3] 435
2 Call Control (CC) [F:3] 439
3 Group Management (GM) [F:3] 449
4 Connectionless TCS (CL) [F:3] 455
5 Supplementary Services (SS) [F:3] 456
6 Message formats [F:3] 459
7 Message coding [F:3] 471
8 Message Error handling [F:3] 487
9 Protocol Parameters [F:3] 489
10 References [F:3] 490
11 List of Figures [F:3] 491
12 List of Tables [F:3] 492
Appendix 1 - TCS Call States [F:3] 493

Part F:4 **Volume 1 (1:2)**

INTEROPERABILITY REQUIREMENTS FOR BLUETOOTH AS A WAP BEARER

Contents **[F:4] 497**

1 Introduction [F:4] 499
2 The Use of WAP In the Bluetooth Environment [F:4] 500
3 WAP Services Overview [F:4] 502
4 WAP in the Bluetooth Piconet [F:4] 506
5 Interoperability Requirements [F:4] 511
6 Service Discovery [F:4] 512
7 References [F:4] 515

Bluetooth.

Part H:1 **Volume 1 (2:2)**

HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

Contents **[H:1] 519**

1 Introduction [H:1] 524
2 Overview of Host Controller Transport Layer [H:1] 528
3 HCI Flow Control [H:1] 529
4 HCI Commands [H:1] 531
5 Events [H:1] 703
6 List of Error Codes [H:1] 745
7 List of Acronyms and Abbreviations [H:1] 755
8 List of Figures [H:1] 756
9 List of Tables [H:1] 757

Part H:2 **Volume 1 (2:2)**

HCI USB TRANSPORT LAYER

Contents **[H:2] 761**

1 Overview [H:2] 762
2 USB Endpoint Expectations [H:2] 764
3 Class Code [H:2] 771
4 Device Firmware Upgrade [H:2] 772
5 Limitations [H:2] 773

Part H:3 **Volume 1 (2:2)**

HCI RS232 TRANSPORT LAYER

Contents **[H:3] 777**

1 General [H:3] 778
2 Overview [H:3] 779
3 Negotiation Protocol [H:3] 780
4 Packet Transfer Protocol [H:3] 784
5 Using delimiters with COBS for synchronization [H:3] 785
6 Using RTS/CTS for Synchronization [H:3] 788
7 References [H:3] 794

Bluetooth.

Part H:4 **Volume 1 (2:2)**

HCI UART TRANSPORT LAYER

Contents **[H:4] 797**

1 General [H:4] 798

2 Protocol [H:4] 799

3 RS232 Settings [H:4] 800

4 Error Recovery [H:4] 801

Part I:1 **Volume 1 (2:2)**

BLUETOOTH TEST MODE

Contents **[I:1] 805**

1 General Description [I:1] 806

2 Test Scenarios [I:1] 808

3 Outline of Proposed LMP Messages [I:1] 817

4 References [I:1] 819

Part I:2 **Volume 1 (2:2)**

BLUETOOTH COMPLIANCE REQUIREMENTS

Contents **[I:2] 823**

1 Scope [I:2] 825

2 Terms Used [I:2] 826

3 Legal Aspects [I:2] 828

4 The Value of the Bluetooth Brand [I:2] 829

5 The Bluetooth Qualification Program [I:2] 830

6 Bluetooth License Requirements for Products [I:2] 832

7 Bluetooth License Provisions for Early Products [I:2] 836

8 Bluetooth Brand License Provisions for Special Products & Marketing [I:2] 837

9 Recommendations Concerning Information about a Product's Bluetooth Capabilities [I:2] 838

10 Quality Management, Configuration Management and Version Control [I:2] 839

11 Appendix A – Example of a "Bluetooth Capability Statement" [I:2] 840

12 Appendix B - Marketing Names of Bluetooth Profiles . [I:2] 841

Bluetooth.

Part I:3 **Volume 1 (2:2)**

TEST CONTROL INTERFACE

Contents **[I:3] 845**

1 Introduction [I:3] 847
2 General Description [I:3] 849
3 Test Configurations [I:3] 854
4 TCI-L2CAP Specification [I:3] 856
5 Abbreviations [I:3] 866

Part K **Profiles - see Volume 2**

Appendix I **Volume 1 (2:2)**

REVISION HISTORY **[@:I] 868**

Appendix II **Volume 1 (2:2)**

CONTRIBUTORS **[@:II] 881**

Appendix III **Volume 1 (2:2)**

ACRONYMS AND ABBREVIATIONS **[@:III] 891**

Appendix IV **Volume 1 (2:2)**

SAMPLE DATA

Contents **[@:IV] 901**

1 Encryption Sample Data [@:IV] 902
2 Frequency Hopping Sample Data—Mandatory Scheme [@:IV] 937
3 Access Code Sample Data [@:IV] 950
4 HEC and Packet Header Sample Data [@:IV] 953
5 CRC Sample Data [@:IV] 954
6 Complete Sample Packets [@:IV] 955
7 Whitening Sequence Sample Data [@:IV] 957
8 FEC Sample Data [@:IV] 960
9 Encryption Key Sample Data [@:IV] 961

Bluetooth.

Appendix V **Volume 1 (2:2)**

BLUETOOTH AUDIO

Contents **[@:V] 987**

1 General Audio Recommendations [@:V] 989

Appendix VI **Volume 1 (2:2)**

BASEBAND TIMERS

Contents **[@:VI] 995**

1 Baseband Timers [@:VI] 996

Appendix VII **Volume 1 (2:2)**

OPTIONAL PAGING SCHEMES

Contents **[@:VII] 1001**

1 General [@:VII] 1003

2 Optional Paging Scheme I [@:VII] 1004

Appendix VIII **Volume 1 (2:2)**

BLUETOOTH ASSIGNED NUMBERS

Contents **[@:VIII] 1011**

1 Bluetooth Baseband [@:VIII] 1012

2 Link Manager Protocol (LMP) [@:VIII] 1018

3 Logical Link Control and Adaptation Protocol [@:VIII] 1019

4 Service Discovery Protocol (SDP) [@:VIII] 1020

5 References [@:VIII] 1028

6 Terms and Abbreviations [@:VIII] 1029

7 List of Figures [@:VIII] 1030

8 List of Tables [@:VIII] 1031

Bluetooth.

Appendix IX **Volume 1 (2:2)**

MESSAGE SEQUENCE CHARTS

Contents **[@:IX] 1035**

1 Introduction [@:IX] 1037

2 Services Without Connection Request [@:IX] 1038

3 ACL Connection Establishment and Detachment [@:IX] 1042

4 Optional Activities After ACL Connection Establishment [@:IX] 1050

5 SCO Connection Establishment and Detachment [@:IX] 1059

6 Special Modes: Sniff, Hold, Park [@:IX] 1062

7 Buffer Management, Flow Control [@:IX] 1068

8 Loopback Mode [@:IX] 1070

9 List of Acronyms and Abbreviations [@:IX] 1073

10 List of Figures [@:IX] 1074

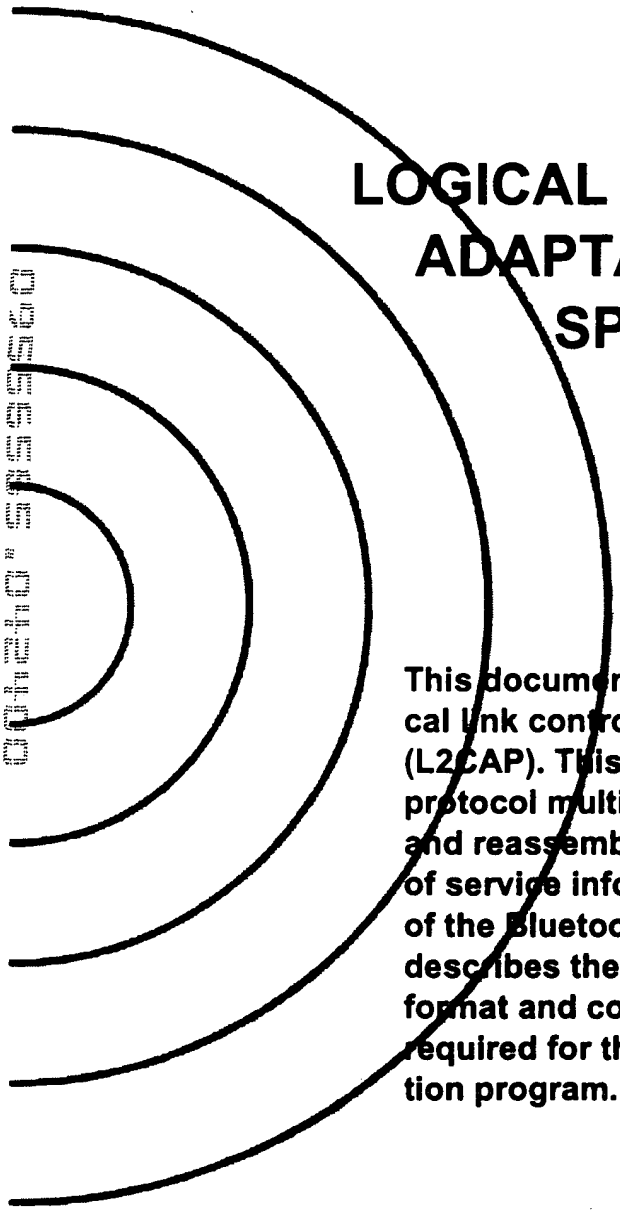
11 List of Tables [@:IX] 1075

12 References [@:IX] 1076

Alphabetical Index **1077**

Part D

LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION



CONFIDENTIAL

This document describes the Bluetooth logical link control and adaptation protocol (L2CAP). This protocol supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. This document is part of the Bluetooth Specification. This document describes the protocol state machine, packet format and composition, and a test interface required for the Bluetooth test and certification program.

CONTENTS

1 Introduction 249
1.1 L2CAP Functional Requirements 250
1.2 Assumptions 252
1.3 Scope 252
2 General Operation 253
2.1 Channel Identifiers 253
2.2 Operation Between Devices 253
2.3 Operation Between Layers 254
2.4 Segmentation and Reassembly 255
2.4.1 Segmentation Procedures 256
2.4.2 Reassembly Procedures 256
3 State Machine 258
3.1 Events 259
3.1.1 Lower-Layer Protocol (LP) to L2CAP events 259
3.1.2 L2CAP to L2CAP Signalling events 260
3.1.3 L2CAP to L2CAP Data events 261
3.1.4 Upper-Layer to L2CAP events 261
3.1.5 Timer events 262
3.2 Actions 263
3.2.1 L2CAP to Lower Layer actions 263
3.2.2 L2CAP to L2CAP Signalling actions 264
3.2.3 L2CAP to L2CAP Data actions 264
3.2.4 L2CAP to Upper Layer actions 264
3.3 Channel Operational States 265
3.4 Mapping Events to Actions 266
4 Data Packet Format 272
4.1 Connection-oriented Channel 272
4.2 Connectionless Data Channel 273
5 Signlling 275
5.1 Command Reject (code 0x01) 277
5.2 Connection Request (code 0x02) 278
5.3 Connection Response (code 0x03) 279
5.4 Configuration Request (code 0x04) 280
5.5 Configure Response (code 0x05) 283
5.6 Disconnection Request (code 0x06) 285
5.7 Disconnection Response (code 0x07) 286
5.8 Echo Request (code 0x08) 286
5.9 Echo Response (code 0x09) 287
5.10 Information Request (CODE 0x0A) 287
5.11 Information Response (CODE 0x0B) 288

6 Configuration Parameter Options 289
6.1 Maximum Transmission Unit (MTU) 289
6.2 Flush Timeout Option 290
6.3 Quality of Service (QoS) Option 291
6.4 Configuration Process 293
6.4.1 Request Path 293
6.4.2 Response Path 294
6.4.3 Configuration State Machine 294
7 Service Primitives 295
7.1 Event Indication 295
7.1.1 L2CA_ConnectInd Callback 296
7.1.2 L2CA_ConfigInd Callback 296
7.1.3 L2CA_DisconnectInd Callback 296
7.1.4 L2CA_QoSViolationInd Callback 296
7.2 Connect 296
7.3 Connect Response 298
7.4 Configure 299
7.5 Configuration Response 301
7.6 Disconnect 302
7.7 Write 303
7.8 Read 304
7.9 Group Create 305
7.10 Group Close 305
7.11 Group Add Member 306
7.12 Group Remove Member 307
7.13 Get Group Membership 308
7.14 Ping 309
7.15 GetInfo 310
7.16 Disable Connectionless Traffic 311
7.17 Enable Connectionless Traffic 312
8 Summary 313
9 References 314
10 List of Figures 315
11 List of Tables 316
Terms and Abbreviations 317
Appendix A: Configuration MSCs 318
Appendix B: Implementation Guidelines 321

1 INTRODUCTION

This section of the Bluetooth Specification defines the Logical Link Control and Adaptation Layer Protocol, referred to as L2CAP. L2CAP is layered over the Baseband Protocol and resides in the data link layer as shown in Figure 1.1. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

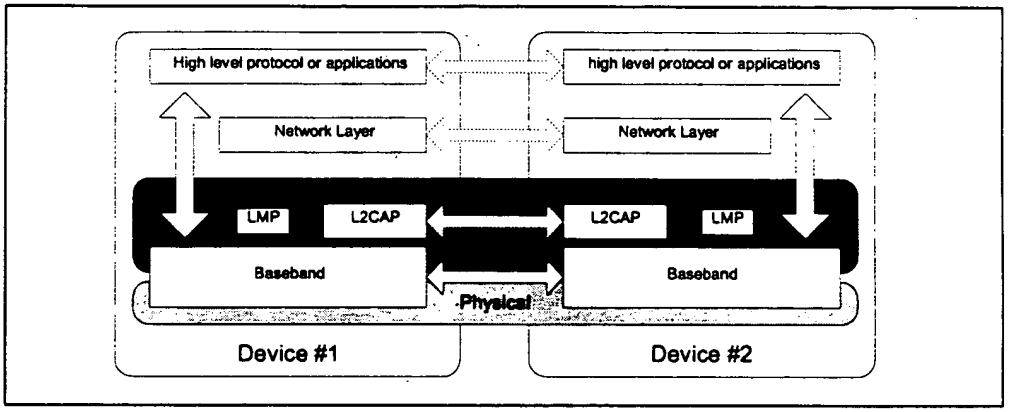


Figure 1.1: L2CAP within protocol layers

The "Baseband Specification" on page 33 defines two link types: Synchronous Connection-Oriented (SCO) links and Asynchronous Connection-Less (ACL) links. SCO links support real-time voice traffic using reserved bandwidth. ACL links support best effort traffic. The L2CAP Specification is defined for only ACL links and no support for SCO links is planned.

For ACL links, use of the AUX1 packet on the ACL link is prohibited. This packet type supports no data integrity checks (no CRC). Because L2CAP depends on integrity checks in the Baseband to protect the transmitted information, AUX1 packets must never be used to transport L2CAP packets.

The format of the ACL payload header is shown below. Figure 1.2 on page 250 displays the payload header used for single-slot packets and Figure 1.3 displays the header used in multi-slot packets. The only difference is the size of the length field. The packet type (a field in the Baseband header) distinguishes single-slot packets from multi-slot packets.

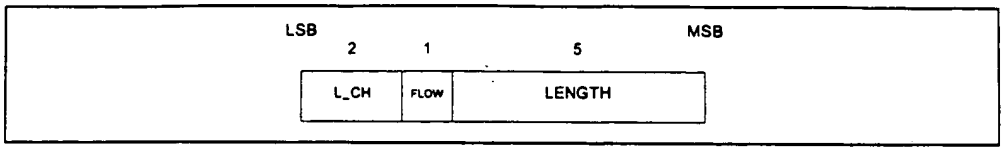


Figure 1.2: ACL Payload Header for single-slot packets

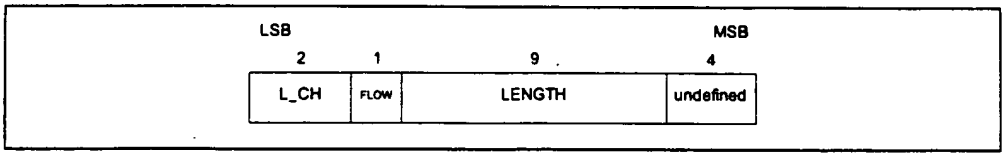


Figure 1.3: ACL Payload Header for multi-slot packets

The 2-bit logical channel (L_CH) field, defined in Table 1.1, distinguishes L2CAP packets from Link Manager Protocol (page 185) packets. The remaining code is reserved for future use.

L_CH code	Logical Channel	Information
00	RESERVED	Reserved for future use
01	L2CAP	Continuation of L2CAP packet
10	L2CAP	Start of L2CAP packet
11	LMP	Link Manager Protocol

Table 1.1: Logical channel L_CH field contents

The FLOW bit in the ACL header is managed by the Link Controller (LC), a Baseband implementation entity, and is normally set to1 ('flow on'). It is set to 0 ('flow off') when no further L2CAP traffic shall be sent over the ACL link. Sending an L2CAP packet with the FLOW bit set to 1 resumes the flow of incoming L2CAP packets. This is described in more detail in "Baseband Specification" on page 33.

1.1 L2CAP FUNCTIONAL REQUIREMENTS

The functional requirements for L2CAP include protocol multiplexing, segmentation and reassembly (SAR), and group management. Figure 1.4 illustrates how L2CAP fits into the Bluetooth Protocol Stack. L2CAP lies above the Baseband Protocol (page 33) and interfaces with other communication protocols such as the Bluetooth Service Discovery Protocol (SDP, page 323), RFCOMM (page 385), and Telephony Control (TCS, page 429). Voice-quality channels for audio and telephony applications are usually run over Baseband SCO links. Packetized audio data, such as IP Telephony, may be sent using communication protocols running over L2CAP.

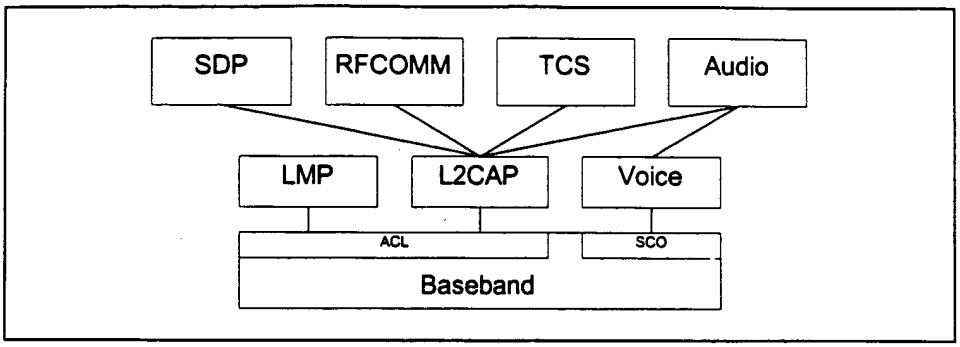


Figure 1.4: L2CAP in Bluetooth Protocol Architecture

Essential protocol requirements for L2CAP include simplicity and low overhead. Implementations of L2CAP must be applicable for devices with limited computational resources. L2CAP should not consume excessive power since that significantly sacrifices power efficiency achieved by the Bluetooth Radio. Memory requirements for protocol implementation should also be kept to a minimum.

The protocol complexity should be acceptable to personal computers, PDAs, digital cellular phones, wireless headsets, joysticks and other wireless devices supported by Bluetooth. Furthermore, the protocol should be designed to achieve reasonably high bandwidth efficiency.

- *Protocol Multiplexing*

 L2CAP must support protocol multiplexing because the Baseband Protocol does not support any 'type' field identifying the higher layer protocol being multiplexed above it. L2CAP must be able to distinguish between upper layer protocols such as the Service Discovery Protocol (page 323), RFCOMM (page 385), and Telephony Control (page 429).

- *Segmentation and Reassembly*

 Compared to other wired physical media, the data packets defined by the Baseband Protocol (page 33) are limited in size. Exporting a maximum transmission unit (MTU) associated with the largest Baseband payload (341 bytes for DH5 packets) limits the efficient use of bandwidth for higher layer protocols that are designed to use larger packets. Large L2CAP packets must be segmented into multiple smaller Baseband packets prior to their transmission over the air. Similarly, multiple received Baseband packets may be reassembled into a single larger L2CAP packet following a simple integrity check (described in Section 2.4.2 on page 256). The Segmentation and Reassembly (SAR) functionality is absolutely necessary to support protocols using packets larger than those supported by the Baseband.

- *Quality of Service*

 The L2CAP connection establishment process allows the exchange of information regarding the quality of service (QoS) expected between two Blue-

tooth units. Each L2CAP implementation must monitor the resources used by the protocol and ensure that QoS contracts are honoured.

- *Groups*

 Many protocols include the concept of a group of addresses. The Baseband Protocol supports the concept of a piconet, a group of devices synchronously hopping together using the same clock. The L2CAP group abstraction permits implementations to efficiently map protocol groups on to piconets. Without a group abstraction, higher level protocols would need to be exposed to the Baseband Protocol and Link Manager functionality in order to manage groups efficiently.

1.2 ASSUMPTIONS

The protocol is designed based on the following assumptions:

1. The ACL link between two units is set up using the Link Manager Protocol (page 185). The Baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than 1 ACL link exists between any two devices.
2. The Baseband always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.
3. L2CAP provides a reliable channel using the mechanisms available at the Baseband layer. The Baseband always performs data integrity checks when requested and resends data until it has been successfully acknowledged or a timeout occurs. Because acknowledgements may be lost, timeouts may occur even after the data has been successfully sent. The Baseband protocol uses a 1-bit sequence number that removes duplicates. Note that the use of Baseband broadcast packets is prohibited if reliability is required since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.

1.3 SCOPE

The following features are outside the scope of L2CAP's responsibilities:

- L2CAP does not transport audio designated for SCO links.
- L2CAP does not enforce a reliable channel or ensure data integrity, that is, L2CAP performs no retransmissions or checksum calculations.
- L2CAP does not support a reliable multicast channel. See Section 4.2.
- L2CAP does not support the concept of a global group name.

2 GENERAL OPERATION

The Logical Link Control and Adaptation Protocol (L2CAP) is based around the concept of *'channels'*. Each one of the end-points of an L2CAP channel is referred to by a *channel identifier.*

2.1 CHANNEL IDENTIFIERS

Channel identifiers (CIDs) are local names representing a logical channel end-point on the device. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. The null identifier (0x0000) is defined as an illegal identifier and must never be used as a destination end-point. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that the same CID is not reused as a local L2CAP channel endpoint for multiple simultaneous L2CAP channels between a local device and some remote device. Table 2.1 summarizes the definition and partitioning of the CID name space.

CID assignment is relative to a particular device and a device can assign CIDs independently from other devices (unless it needs to use any of the reserved CIDs shown in the table below). Thus, even if the same CID value has been assigned to (remote) channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device.

CID	Description
0x0000	Null identifier
0x0001	Signalling channel
0x0002	Connectionless reception channel
0x0003-0x003F	Reserved
0x0040-0xFFFF	Dynamically allocated

Table 2.1: CID Definitions

2.2 OPERATION BETWEEN DEVICES

Figure 2.1 on page 254 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel 'group' where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signalling channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of these channels. Support for a signalling chan-

nel within an L2CAP entity is mandatory. Another CID is reserved for all incoming connectionless data traffic. In the example below, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this channel ID is directed to the remote channel reserved for connectionless data traffic.

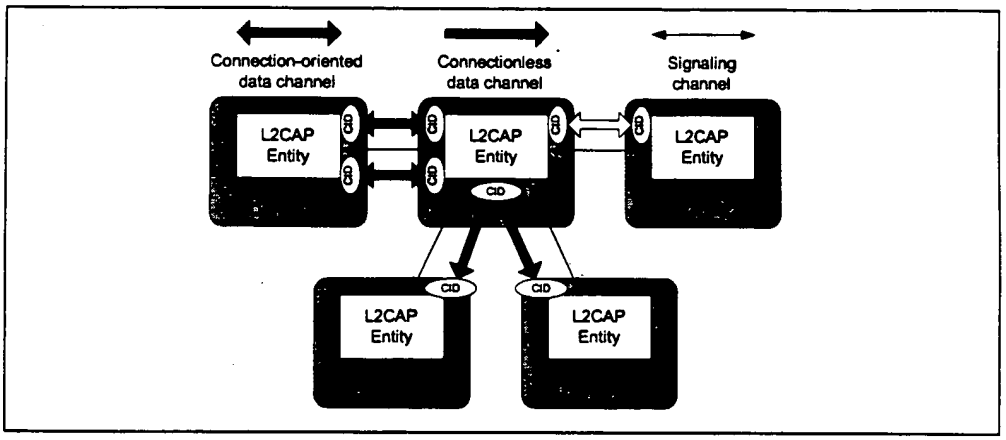


Figure 2.1: Channels between devices

Table 2.2 describes the various channels and their source and destination identifiers. An 'allocated' channel is created to represent the local endpoint and should be in the range 0x0040 to 0xFFFF. Section 3 on page 258 describes the state machine associated with each connectionless channel. Section 4.1 on page 272 describes the packet format associated with bi-directional channels and Section 4.2 on page 273 describes the packet format associated with unidirectional channels.

Channel Type	Local CID	Remote CID
Connection-oriented	Dynamically allocated	Dynamically allocated
Connectionless data	Dynamically allocated	0x0002 (fixed)
Signalling	0x0001 (fixed)	0x0001 (fixed)

Table 2.2: Types of Channel Identifiers

2.3 OPERATION BETWEEN LAYERS

L2CAP implementations should follow the general architecture described below. L2CAP implementations must transfer data between higher layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation must also support a set of signalling commands for use between L2CAP implementations. L2CAP implementations should also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is an implementation-dependent process.

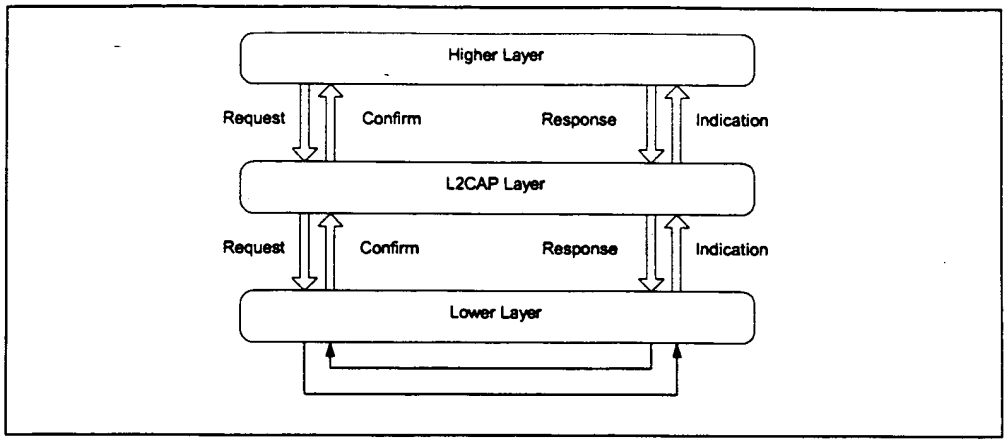


Figure 2.2: L2CAP Architecture

2.4 SEGMENTATION AND REASSEMBLY

Segmentation and reassembly (SAR) operations are used to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest Baseband packet. This reduces overhead by spreading the network and transport packets used by higher layer protocols over several Baseband packets. All L2CAP packets may be segmented for transfer over Baseband packets. The protocol does not perform any segmentation and reassembly operations but the packet format supports adaptation to smaller physical frame sizes. An L2CAP implementation exposes the outgoing (i.e., the remote host's receiving) MTU and segments higher layer packets into 'chunks' that can be passed to the Link Manager via the Host Controller Interface (HCI), whenever one exists. On the receiving side, an L2CAP implementation receives 'chunks' from the HCI and reassembles those chunks into L2CAP packets using information provided through the HCI and from the packet header.

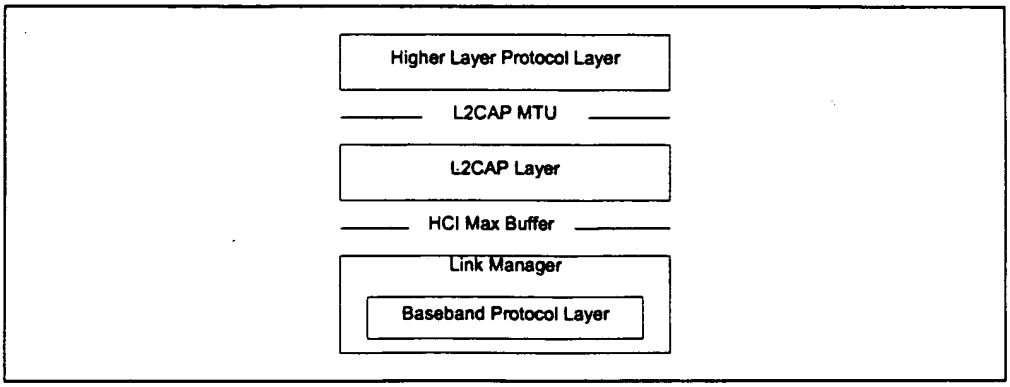


Figure 2.3: L2CAP SAR Variables

Segmentation and Reassembly is implemented using very little overhead in Baseband packets. The two L_CH bits defined in the first byte of Baseband

payload (also called the frame header) are used to signal the start and continuation of L2CAP packets. L_CH shall be '10' for the first segment in an L2CAP packet and '01' for a continuation segment. An example use of SAR is shown in Figure 2.4.

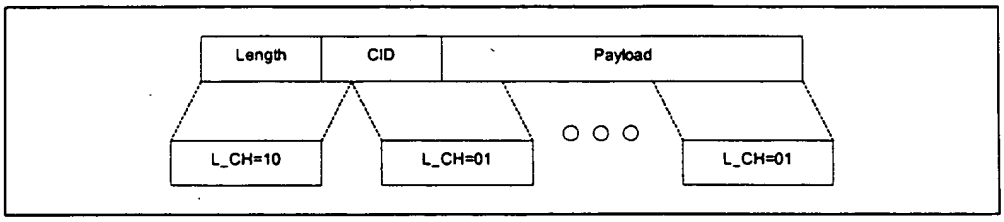


Figure 2.4: L2CAP segmentation

2.4.1 Segmentation Procedures

The L2CAP maximum transmission unit (MTU) will be exported using an implementation specific service interface. It is the responsibility of the higher layer protocol to limit the size of packets sent to the L2CAP layer below the MTU limit. An L2CAP implementation will segment the packet into protocol data units (PDUs) to send to the lower layer. If L2CAP runs directly over the Baseband Protocol, an implementation may segment the packet into Baseband packets for transmission over the air. If L2CAP runs above the host controller interface (typical scenario), an implementation may send block-sized chunks to the host controller where they will be converted into Baseband packets. All L2CAP segments associated with an L2CAP packet must be passed through to the Baseband before any other L2CAP packet destined to the same unit may be sent.

2.4.2 Reassembly Procedures

The Baseband Protocol delivers ACL packets in sequence and protects the integrity of the data using a 16-bit CRC. The Baseband also supports reliable connections using an automatic repeat request (ARQ) mechanism. As the Baseband controller receives ACL packets, it either signals the L2CAP layer on the arrival of each Baseband packets, or accumulates a number of packets before the receive buffer fills up or a timer expires before signalling the L2CAP layer.

L2CAP implementations must use the length field in the header of L2CAP packets, see Section 4 on page 272, as a consistency check and discard any L2CAP packets that fail to match the length field. If channel reliability is not needed, packets with improper lengths may be silently discarded. For reliable channels, L2CAP implementations must indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as specified in Section 6.2 on page 290.

Figure 2.5 on page 257 illustrates the use of segmentation and reassembly operations to transmit a single higher layer PDU. Note that while there is a one-to-one mapping between a high layer PDU and an L2CAP packet, the segment

size used by the segmentation and reassembly routines is left to the implementation and may differ from the sender to the receiver.

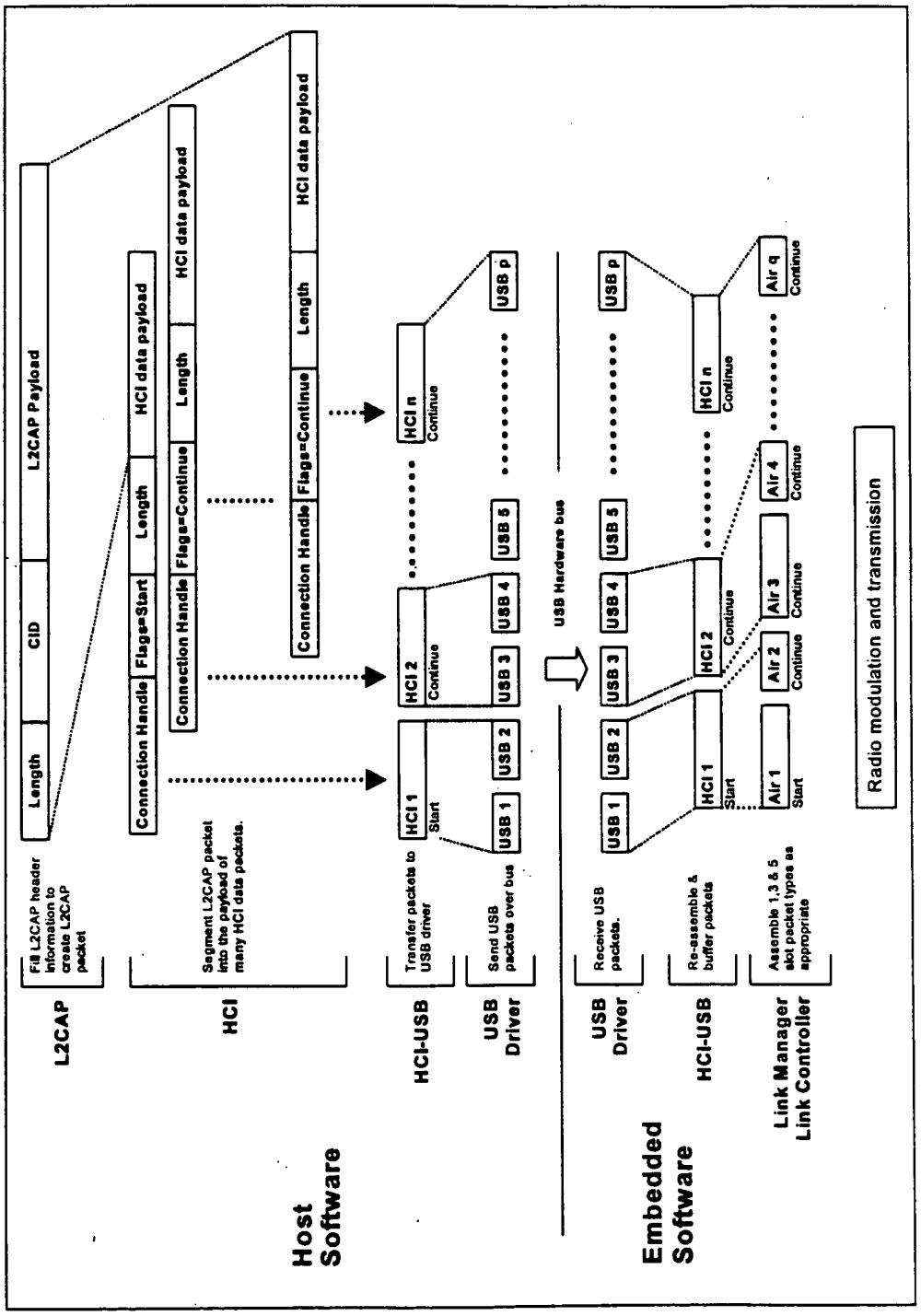


Figure 2.5: Segmentation and Reassembly Services in a unit with an HCI[1]

3 STATE MACHINE

This section describes the L2CAP connection-oriented channel state machine. The section defines the states, the events causing state transitions, and the actions to be performed in response to events. This state machine is only pertinent to bi-directional CIDs and is not representative of the signalling channel or the uni-directional channel.

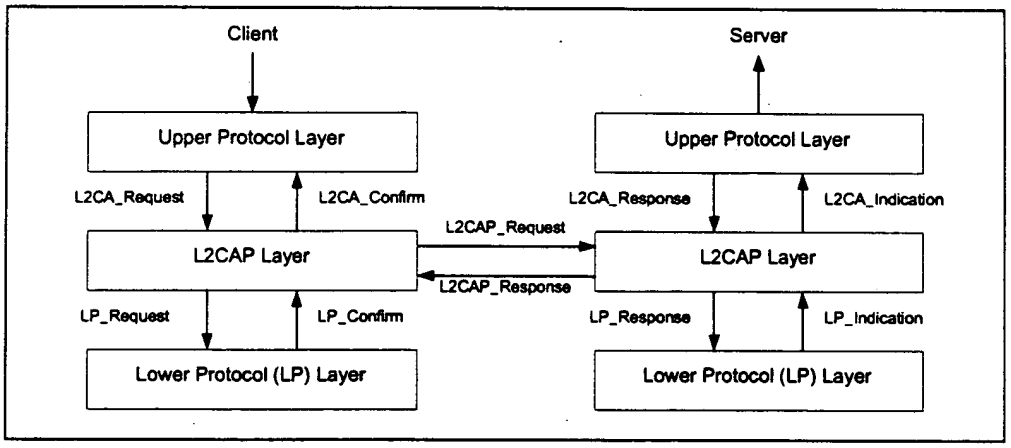


Figure 3.1: L2CAP Layer Interactions

Figure 3.1 illustrates the events and actions performed by an implementation of the L2CAP layer. Client and Server simply represent the initiator of the request and the acceptor of the request respectively. An application-level Client would both initiate and accept requests. The naming convention is as follows. The interface between two layers (vertical interface) uses the prefix of the lower layer offering the service to the higher layer, e.g., L2CA. The interface between two entities of the same layer (horizontal interface) uses the prefix of the protocol (adding a P to the layer identification), e.g., L2CAP. Events coming from above are called Requests (Req) and the corresponding replies are called Confirms (Cfm). Events coming from below are called Indications (Ind) and the corresponding replies are called Responses (Rsp). Responses requiring further processing are called Pending (Pnd). The notation for Confirms and Responses assumes positive replies. Negative replies are denoted by a 'Neg' suffix such as L2CAP_ConnectCfmNeg.

While Requests for an action always result in a corresponding Confirmation (for the successful or unsuccessful satisfaction of the action), Indications do not always result into corresponding Responses. The latter is especially true, if the Indications are informative about locally triggered events, e.g., seeing the

1. For simplicity, the stripping of any additional HCI and USB specific information fields prior to the creation of the baseband packets (Air_1, Air_2, etc.) is not shown in the figure.

LP_QoSViolationInd in Section 3.1.1 on page 259, or *L2CA_TimeOutInd* in Section 3.2.4 on page 264.

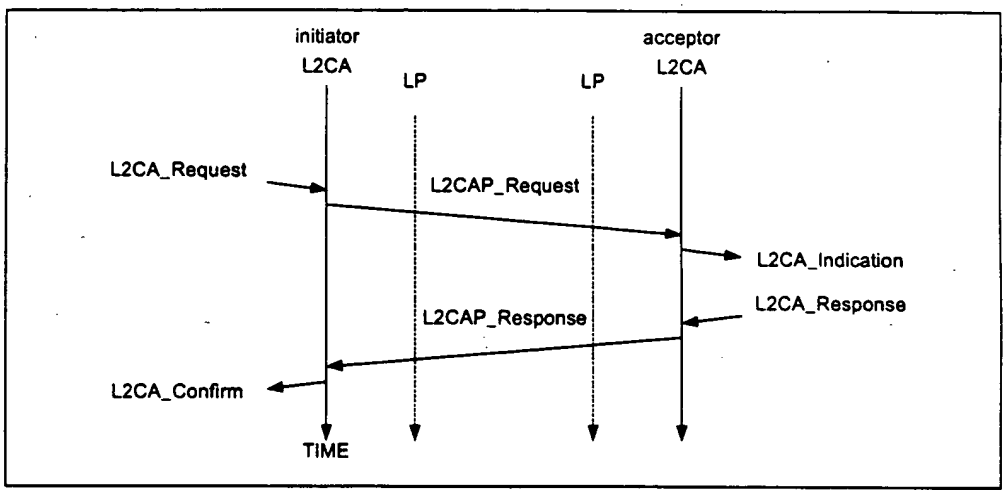


Figure 3.2: MSC of Layer Interactions

Figure 3.2 uses a message sequence chart (MSC) to illustrate the normal sequence of events. The two outer vertical lines represent the L2CA interface on the initiator (the device issuing a request) and the acceptor (the device responding to the initiator's request). Request commands at the L2CA interface result in Requests defined by the protocol. When the protocol communicates the request to the acceptor, the remote L2CA entity presents the upper protocol with an Indication. When the acceptor's upper protocol responds, the response is packaged by the protocol and communicated back the to initiator. The result is passed back to the initiator's upper protocol using a Confirm message.

3.1 EVENTS

Events are all incoming messages to the L2CA layer along with timeouts. Events are partitioned into five categories: Indications and Confirms from lower layers, Requests and Responses from higher layers, data from peers, signal Requests and Responses from peers, and events caused by timer expirations.

3.1.1 Lower-Layer Protocol (LP) to L2CAP events

- *LP_ConnectCfm*

 Confirms the request (see *LP_ConnectReq* in Section 3.2.1) to establish a lower layer (Baseband) connection. This includes passing the authentication challenge if authentication is required to establish the physical link.

- *LP_ConnectCfmNeg*

 Confirms the failure of the request (see *LP_ConnectReq* in Section 3.2.1) to establish a lower layer (Baseband) connection failed. This could be because

the device could not be contacted, refused the request, or the LMP authentication challenge failed.

- *LP_ConnectInd*

 Indicates the lower protocol has successfully established connection. In the case of the Baseband, this will be an ACL link. An L2CAP entity may use to information to keep track of what physical links exist.

- *LP_DisconnectInd*

 Indicates the lower protocol (Baseband) has been shut down by LMP commands or a timeout event.

- *LP_QoSCfm*

 Confirms the request (see *LP_QoSReq* in Section 3.2.1) for a given quality of service.

- *LP_QoSCfmNeg*

 Confirms the failure of the request (see *LP_QoSReq* in Section 3.2.1) for a given quality of service.

- *LP_QoSViolationInd*

 Indicates the lower protocol has detected a violation of the QoS agreement specified in the previous *LP_QoSReq* (see Section 3.2.1).

3.1.2 L2CAP to L2CAP Signalling events

L2CAP to L2CAP signalling events are generated by each L2CAP entity following the exchange of the corresponding L2CAP signalling PDUs, see Section 5. L2CAP signalling PDUs, like any other L2CAP PDUs, are received from a lower layer via a lower protocol indication event. For simplicity of the presentation, we avoid a detailed description of this process, and we assume that signalling events are exchanged directly between the L2CAP peer entities as shown in Figure 3.1 on page 258.

- *L2CAP_ConnectReq*

 A Connection Request packet has been received.

- *L2CAP_ConnectRsp*

 A Connection Response packet has been received with a positive result indicating that the connection has been established.

- *L2CAP_ConnectRspPnd*

 A Connection Response packet has been received indicating the remote endpoint has received the request and is processing it.

- *L2CAP_ConnectRspNeg*

 A Connection Response packet has been received, indicating that the connection could not be established.

- *L2CAP_ConfigReq*

A Configuration Request packet has been received indicating the remote endpoint wishes to engage in negotiations concerning channel parameters.

- *L2CAP_ConfigRsp*

 A Configuration Response packet has been received indicating the remote endpoint agrees with all the parameters being negotiated.

- *L2CAP_ConfigRspNeg*

 A Configuration Response packet has been received indicating the remote endpoint does not agree to the parameters received in the response packet.

- *L2CAP_DisconnectReq*

 A Disconnection Request packet has been received and the channel must initiate the disconnection process. Following the completion of an L2CAP channel disconnection process, an L2CAP entity should return the corresponding local CID to the pool of 'unassigned' CIDs.

- *L2CAP_DisconnectRsp*

 A Disconnection Response packet has been received. Following the receipt of this signal, the receiving L2CAP entity may return the corresponding local CID to the pool of unassigned CIDs. There is no corresponding negative response because the Disconnect Request must succeed.

3.1.3 L2CAP to L2CAP Data events

- *L2CAP_Data*

 A Data packet has been received.

3.1.4 Upper-Layer to L2CAP events

- *L2CA_ConnectReq*

 Request from upper layer for the creation of a channel to a remote device.

- *L2CA_ConnectRsp*

 Response from upper layer to the indication of a connection request from a remote device (see L2CA_ConnectInd in Section 3.2.4).

- *L2CA_ConnectRspNeg*

 Negative response (rejection) from upper layer to the indication of a connection request from a remote device (see *L2CA_ConnectInd* in Section 3.2.4).

- *L2CA_ConfigReq*

 Request from upper layer to (re)configure the channel.

- *L2CA_ConfigRsp*

 Response from upper layer to the indication of a (re) configuration request (see *L2CA_ConfigInd* in Section 3.2.4).

- *L2CA_ConfigRspNeg*

A negative response from upper layer to the indication of a (re) configuration request (see *L2CA_ConfigInd* in Section 3.2.4).

- *L2CA_DisconnectReq*

 Request from upper layer for the immediate disconnection of a channel.

- *L2CA_DisconnectRsp*

 Response from upper layer to the indication of a disconnection request (see *L2CA_DisconnectInd* in Section 3.2.4). There is no corresponding negative response, the disconnect indication must always be accepted.

- *L2CA_DataRead*

 Request from upper layer for the transfer of received data from L2CAP entity to upper layer.

- *L2CA_DataWrite*

 Request from upper layer for the transfer of data from the upper layer to L2CAP entity for transmission over an open channel.

3.1.5 Timer events

- *RTX*

 The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signalling requests. This timer is started when a signalling request (see Section 5 on page 275) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value must be reset to a new value at least double the previous value.

 Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before disconnecting the channel. The decision should be based on the flush timeout of the signalling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level. For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level.

 The value of this timer is implementation-dependent but the minimum initial value is 1 second and the maximum initial value is 60 seconds. One RTX timer MUST exist for each outstanding signalling request, including each Echo Request. The timer disappears on the final expiration, when the response is received, or the physical link is lost. The maximum elapsed time between the initial start of this timer and the initiation of channel disconnection (if no response is received) is 60 seconds.

- *ERTX*

 The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing addi-

tional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an *L2CAP_ConnectRspPnd* event is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be disconnected. If a duplicate Request is sent, the particular ERTX timer disappears, replaced by a new RTX timer and the whole timing procedure restarts as described previously for the RTX timer.

The value of this timer is implementation-dependent but the minimum initial value is 60 seconds and the maximum initial value is 300 seconds. Similar to RTX, there MUST be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel disconnection (if no response is received) is 300 seconds.

3.2 ACTIONS

Actions are partitioned into five categories: Confirms and Indications to higher layers, Request and Responses to lower layers, Requests and Responses to peers, data transmission to peers, and setting timers.

3.2.1 L2CAP to Lower Layer actions

- *LP_ConnectReq*

 L2CAP requests the lower protocol to create a connection. If a physical link to the remote device does not exist, this message must be sent to the lower protocol to establish the physical connection. Since no more than a single ACL link between two devices is assumed, see Section 1.2 on page 252, additional L2CAP channels between these two devices must share the same baseband ACL link.
 Following the processing of the request, the lower layer returns with an *LP_ConnectCfm* or an *LP_ConnectCfmNeg* to indicate whether the request has been satisfied or not, respectively.

- *LP_QoSReq*

 L2CAP requests the lower protocol to accommodate a particular QoS parameter set. Following the processing of the request, the lower layer returns with an *LP_QoSCfm* or an *LP_QoSCfmNeg* to indicate whether the request has been satisfied or not, respectively

- *LP_ConnectRsp*

 A positive response accepting the previous connection indication request (see *LP_ConnectInd* in Section 3.1.1).

- *LP_ConnectRspNeg*

 A negative response denying the previous connection indication request (see *LP_ConnectInd* in Section 3.1.1).

3.2.2 L2CAP to L2CAP Signalling actions

This section contains the same names identified in Section 3.1.2 except the actions refer to the transmission, rather than reception, of these messages.

3.2.3 L2CAP to L2CAP Data actions

This section is the counterpart of Section 3.1.3. Data transmission is the action performed here.

3.2.4 L2CAP to Upper Layer actions

- *L2CA_ConnectInd*

 Indicates a Connection Request has been received from a remote device (see *L2CA_ConnectReq* in Section 3.1.4).

- *L2CA_ConnectCfm*

 Confirms that a Connection Request has been accepted (see (*L2CAP_ConnectReq* in Section 3.1.4) following the receipt of a Connection message from the remote device.

- *L2CA_ConnectCfmNeg*

 Negative confirmation (failure) of a Connection Request (see *L2CA_ConnectReq* in Section 3.1.4). An RTX timer expiration (see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

- *L2CA_ConnectPnd*

 Confirms that a Connection Response (pending) has been received from the remote device.

- *L2CA_ConfigInd*

 Indicates a Configuration Request has been received from a remote device.

- *L2CA_ConfigCfm*

 Confirms that a Configuration Request has been accepted (see *L2CA_ConfigReq* in Section 3.1.4) following the receipt of a Configuration Response from the remote device.

- *L2CA_ConfigCfmNeg*

 Negative confirmation (failure) of a Configuration Request (see *L2CA_ConfigReq* in Section 3.1.4). An RTX timer expiration (see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

- *L2CA_DisconnectInd*

 Indicates a Disconnection Request has been received from a remote device or the remote device has been disconnected because it has failed to respond to a signalling request. See Section 3.1.5

- *L2CA_DisconnectCfm*

 Confirms that a Disconnect Request has been processed by the remote device (see *L2CA_DisconnectReq* in Section 3.1.4) following the receipt of a Disconnection Response from the remote device. An RTX timer expiration (see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Disconnect Request can substitute for a Disconnect Response and result in this action. Upon receiving this event the upper layer knows the L2CAP channel has been terminated. There is no corresponding negative confirm.

- *L2CA_TimeOutInd*

 Indicates that a RTX or ERTX timer has expired. This indication will occur an implementation-dependant number of times before the L2CAP implementation will give up and send a L2CA_DisconnectInd.

- *L2CA_QoSViolationInd*

 Indicates that the quality of service agreement has been violated.

3.3 CHANNEL OPERATIONAL STATES

- *CLOSED*

 In this state, there is no channel associated with this CID. This is the only state when a link level connection (Baseband) may not exist. Link disconnection forces all other states into the CLOSED state.

- *W4_L2CAP_CONNECT_RSP*

 In this state, the CID represents a local end-point and an L2CAP_ConnectReq message has been sent referencing this endpoint and it is now waiting for the corresponding L2CAP_ConnectRsp message.

- *W4_L2CA_CONNECT_RSP*

 In this state, the remote end-point exists and an *L2CAP_ConnectReq* has been received by the local L2CAP entity. An L2CA_ConnectInd has been sent to the upper layer and the part of the local L2CAP entity processing the received *L2CAP_ConnectReq* waits for the corresponding response. The response may require a security check to be performed.

- *CONFIG*

 In this state, the connection has been established but both sides are still negotiating the channel parameters. The Configuration state may also be entered when the channel parameters are being renegotiated. Prior to entering the CONFIG state, all outgoing data traffic should be suspended since the traffic parameters of the data traffic are to be renegotiated. Incoming data traffic must be accepted until the remote channel endpoint has entered the CONFIG state.

In the CONFIG state, both sides must issue L2CAP_ConfigReq messages – if only defaults are being used, a null message should be sent, see Section 5.4 on page 280. If a large amount of parameters need to be negotiated, multiple messages may be sent to avoid any MTU limitations and negotiate incrementally – see Section 6 on page 289 for more details.

Moving from the CONFIG state to the OPEN state requires both sides to be ready. An L2CAP entity is ready when it has received a positive response to its final request and it has positively responded to the final request from the remote device.

- *OPEN*

 In this state, the connection has been established and configured, and data flow may proceed.

- *W4_L2CAP_DISCONNECT_RSP*

 In this state, the connection is shutting down and an L2CAP_DisconnectReq message has been sent. This state is now waiting for the corresponding response.

- *W4_L2CA_DISCONNECT_RSP*

 In this state, the connection on the remote endpoint is shutting down and an L2CAP_DisconnectReq message has been received. An L2CA_DisconnectInd has been sent to the upper layer to notify the owner of the CID that the remote endpoint is being closed. This state is now waiting for the corresponding response from the upper layer before responding to the remote endpoint.

3.4 MAPPING EVENTS TO ACTIONS

Table 3.1 defines the actions taken in response to events that occur in a particular state. Events that are not listed in the table, nor have actions marked N/C (for no change), are assumed to be errors and silently discarded.

Data input and output events are only defined for the Open and Configuration states. Data may not be received during the initial Configuration state, but may be received when the Configuration state is re-entered due to a reconfiguration process. Data received during any other state should be silently discarded.

Event	Current State	Action	New State
LP_ConnectCfm	CLOSED	Flag physical link as up and initiate the L2CAP connection.	CLOSED
LP_ConnectCfmNeg	CLOSED	Flag physical link as down and fail any outstanding service connection requests by sending an L2CA_ConnectCfmNeg message to the upper layer.	CLOSED
LP_ConnectInd	CLOSED	Flag link as up.	CLOSED
LP_DisconnectInd	CLOSED	Flag link as down.	CLOSED
LP_DisconnectInd	Any except CLOSED	Send upper layer L2CA_DisconnectInd message.	CLOSED
LP_QoSViolationInd	Any but OPEN	Discard	N/C
LP_QoSViolationInd	OPEN	Send upper layer L2CA_QoSViolationInd message. If service level is guaranteed, terminate the channel.	OPEN or W4_L2CA_DISCONNECT_RSP
L2CAP_ConnectReq	CLOSED (CID dynamically allocated from free pool.)	Send upper layer L2CA_ConnectInd. Optionally: Send peer L2CAP_ConnectRspPnd	W4_L2CA_CONNECT_RSP
L2CAP_ConnectRsp	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectCfm message. Disable RTX timer.	CONFIG
L2CAP_ConnectRspPnd	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectPnd message. Disable RTX timer and start ERTX timer.	N/C
L2CAP_ConnectRspNeg	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectCfmNeg message. Return CID to free pool. Disable RTX/ERTX timers.	CLOSED
L2CAP_ConfigReq	CLOSED	Send peer L2CAP_ConfigRspNeg message.	N/C
L2CAP_ConfigReq	CONFIG	Send upper layer L2CA_ConfigInd message.	N/C

Table 3.1: L2CAP Channel State Machine

Event	Current State	Action	New State
L2CAP_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send upper layer L2CA_ConfigInd message.	CONFIG
L2CAP_ConfigRsp	CONFIG	Send upper layer L2CA_ConfigCfm message. Disable RTX timer. If an L2CAP_ConfigReq message has been received and positively responded to, then enter OPEN state, otherwise remain in CONFIG state.	N/C or OPEN
L2CAP_ConfigRsp Neg	CONFIG	Send upper layer L2CA_ConfigCfmNeg message. Disable RTX timer.	N/C
L2CAP_Disconnect Req	CLOSED	Send peer L2CAP_DisconnectRsp message.	N/C
L2CAP_Disconnect Req	Any except CLOSED	Send upper layer L2CA_DisconnectInd message.	W4_L2CA_DISCONNECT_RSP
L2CAP_Disconnect Rsp	W4_L2CAP_DISCONNECT_RSP	Send upper layer L2CA_DisconnectCfm message. Disable RTX timer.	CLOSED
L2CAP_Data	OPEN or CONFIG	If complete L2CAP packet received, send upper layer L2CA_Read confirm.	N/C
L2CA_ConnectReq	CLOSED (CID dynamically allocated from free pool)	Send peer LP2CAP_ConnectReq message. Start RTX timer.	W4_L2CAP_CONNECT_RSP
L2CA_ConnectRsp	W4_L2CA_CONNECT_RSP	Send peer L2CAP_ConnectRsp message.	CONFIG
L2CA_ConnectRsp Neg	W4_L2CA_CONNECT_RSP	Send peer L2CAP_ConnectRspNeg message. Return CID to free pool.	CLOSED
L2CA_ConfigReq	CLOSED	Send upper layer L2CA_ConfigCfmNeg message.	N/C
L2CA_ConfigReq	CONFIG	Send peer L2CAP_ConfigReq message. Start RTX timer.	N/C

Table 3.1: L2CAP Channel State Machine

Event	Current State	Action	New State
L2CA_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send peer L2CAP_ConfigReq message. Start RTX timer.	CONFIG
L2CA_ConfigRsp	CONFIG	Send peer L2CAP_ConfigRsp message. If all outstanding L2CAP_ConfigReq messages have received positive responses then move in OPEN state. Otherwise, remain in CONFIG state.	N/C or OPEN
L2CA_ConfigRspNeg	CONFIG	Send peer L2CAP_ConfigRspNeg message.	N/C
L2CA_Disconnect Req	OPEN or CONFIG	Send peer L2CAP_DisconnectReq message. Start RTX timer.	W4_L2CAP_ DISCONNECT _RSP
L2CA_DisconnectRsp	W4_L2CA_ DISCONNECT_ RSP	Send peer L2CAP_DisconnectRsp message. Return CID to free pool.	CLOSED
L2CA_DataRead	OPEN	If payload complete, transfer payload to InBuffer.	OPEN
L2CA_DataWrite	OPEN	Send peer L2CAP_Data message.	OPEN
Timer_RTX	Any	Send upper layer L2CA_TimeOutInd message. If final expiration, return CID to free pool else re-send Request.	CLOSED
Timer_ERTX	Any	Send upper layer L2CA_TimeOutInd message. If final expiration, return CID to free pool else re-send Request.	CLOSED

Table 3.1: L2CAP Channel State Machine

Figure 3.3 illustrates a simplified state machine and typical transition path taken by an initiator and acceptor. The state machine shows what events cause state transitions and what actions are also taken while the transitions occur. Not all the events listed in Table 3.1 are included in the simplified State Machine to avoid cluttering the figure.

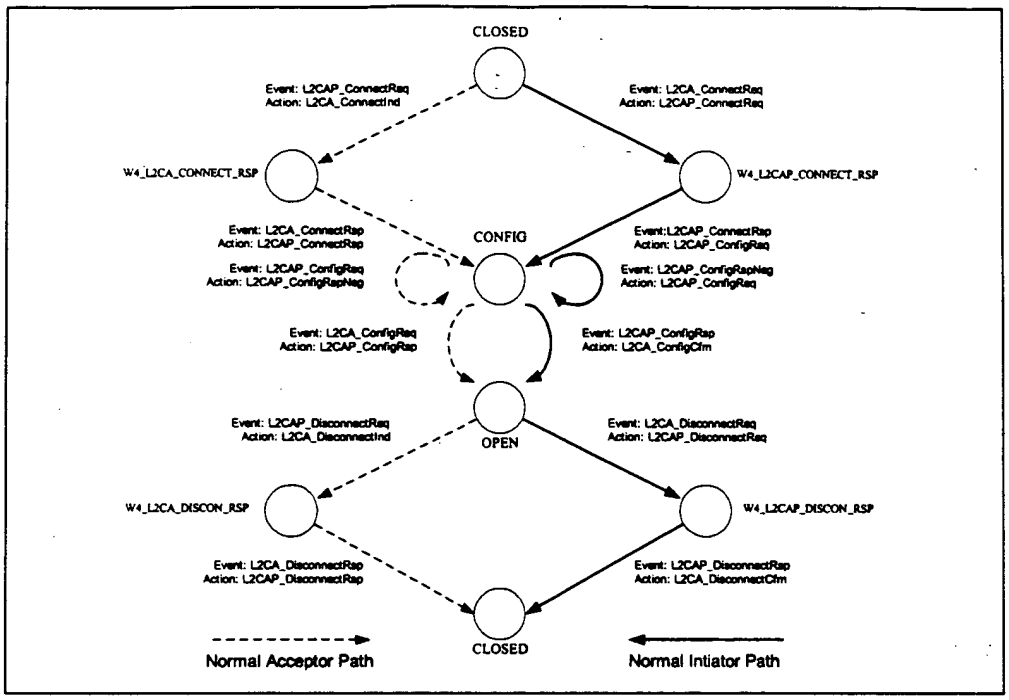


Figure 3.3: State Machine Example

Figure 3.4 presents another illustration of the events and actions based around the messages sequences being communicated between two devices. In this example, the initiator is creating the first L2CAP channel between two devices. Both sides start in the CLOSED state. After receiving the request from the upper layer, the entity requests the lower layer to establish a physical link. If no physical link exists, LMP commands are used to create the physical link between the devices. Once the physical link is established, L2CAP signals may be sent over it.

Figure 3.4 is an example and not all setup sequences will be identical to the one illustrated below.

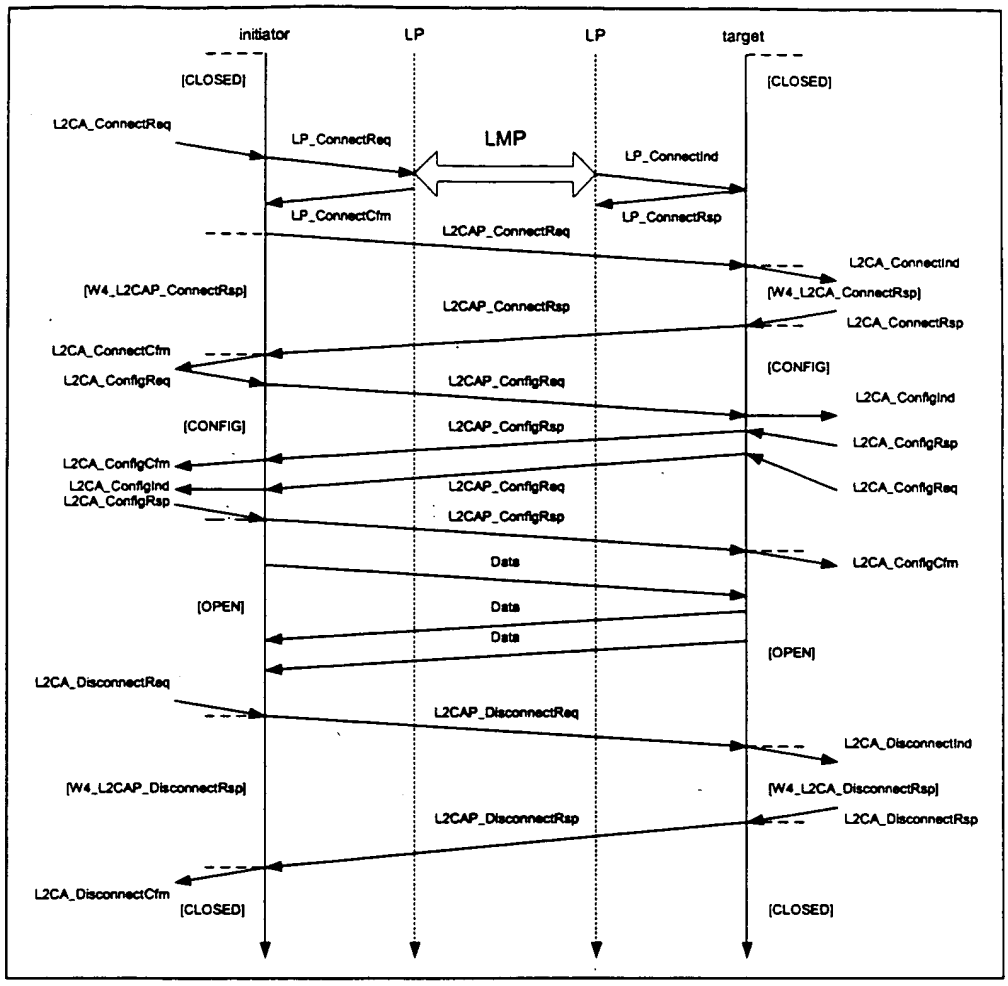


Figure 3.4: Message Sequence Chart of Basic Operation

4 DATA PACKET FORMAT

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless.
All packet fields use Little Endian byte order.

4.1 CONNECTION-ORIENTED CHANNEL

Figure 4.1 illustrates the format of the L2CAP packet (also referred to as the L2CAP PDU) within a connection-oriented channel.

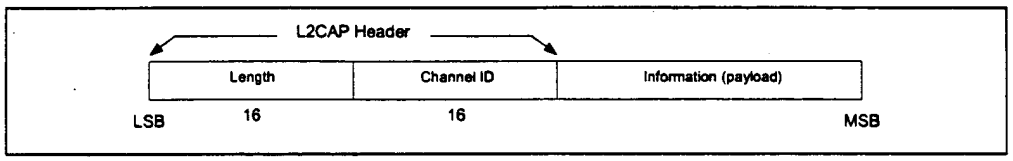


Figure 4.1: L2CAP Packet (field sizes in bits)

The fields shown are:

- *Length: 2 octets (16 bits)*

Length indicates the size of information payload in bytes, excluding the length of the L2CAP header. The length of an information payload can be up to 65535 bytes. The Length field serves as a simple integrity check of the reassembled L2CAP packet on the receiving end.

- *Channel ID: 2 octets*

 The channel ID identifies the destination channel endpoint of the packet. The scope of the channel ID is relative to the device the packet is being sent to.

- *Information: 0 to 65535 octets*

 This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The minimum supported MTU for connection-oriented packets (MTU_{cno}) is negotiated during channel configuration (see Section 6.1 on page 289). The minimum supported MTU for the signalling packet (MTU_{sig}) is 48 bytes (see Section 5 on page 275).

4.2 CONNECTIONLESS DATA CHANNEL

In addition to connection-oriented channels, L2CAP also exports the concept of a group-oriented channel. Data sent to the 'group' channel is sent to all members of the group in a best-effort manner. Groups have no quality of service associated with them. Group channels are unreliable; L2CAP makes no guarantee that data sent to the group successfully reaches all members of the group. If reliable group transmission is required, it must be implemented at a higher layer.

Transmissions to a group must be non-exclusively sent to all members of that group. The local device cannot be a member of the group, and higher layer protocols are expected to loopback any data traffic being sent to the local device. Non-exclusive implies non-group members may receive group transmissions and higher level (or link level) encryption can be used to support private communication.

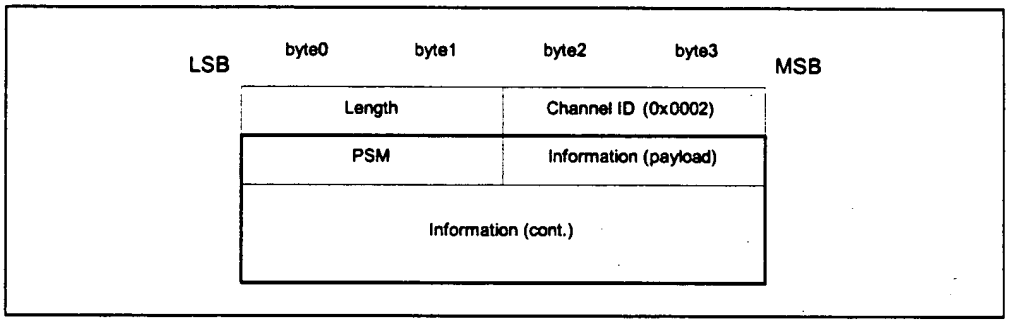


Figure 4.2: Connectionless Packet

The fields shown are:

- *Length: 2 octets*

 Length indicates the size of information payload plus the PSM field in bytes, excluding the length of the L2CAP header.

- *Channel ID: 2 octets*

 Channel ID (0x0002) reserved for connectionless traffic.

- *Protocol/Service Multiplexer (PSM): 2 octets (minimum)*

 The PSM field is based on the ISO 3309 extension mechanism for address fields. All content of the PSM field, referred to as the PSM value, must be ODD, that is, the least significant bit of the least significant octet must be '1'. Also, all PSM values must be assigned such that the least significant bit of the most significant octet equals '0'. This allows the PSM field to be extended beyond 16 bits. The PSM value definitions are specific to L2CAP and assigned by the Bluetooth SIG. For more information on the PSM field see Section 5.2 on page 278.

- *Information: 0 to 65533 octets*

 The payload information to be distributed to all members of the group. Implementations must support a minimum connectionless MTU (MTU_{cnl}) of 670 octets, unless explicitly agreed upon otherwise, e.g., for single operation devices that are built to comply to a specific Bluetooth profile that dictates the use of a specific MTU for connectionless traffic that is less than MTU_{cnl}.

The L2CAP group service interface provides basic group management mechanisms including creating a group, adding members to a group, and removing members from a group. There are no pre-defined groups such as 'all radios in range'.

5 SIGNALLING

This section describes the signalling commands passed between two L2CAP entities on remote devices. All signalling commands are sent to CID 0x0001. The L2CAP implementation must be able to determine the Bluetooth address (BD_ADDR) of the device that sent the commands. Figure 5.1 illustrates the general format of all L2CAP packets containing signalling commands. Multiple commands may be sent in a single (L2CAP) packet and packets are sent to CID 0x0001. MTU Commands take the form of Requests and Responses. All L2CAP implementations must support the reception of signalling packets whose MTU (MTU_{sig}) does not exceed 48 bytes. L2CAP implementations should not use signalling packets beyond this size without first testing whether the implementation can support larger signalling packets.

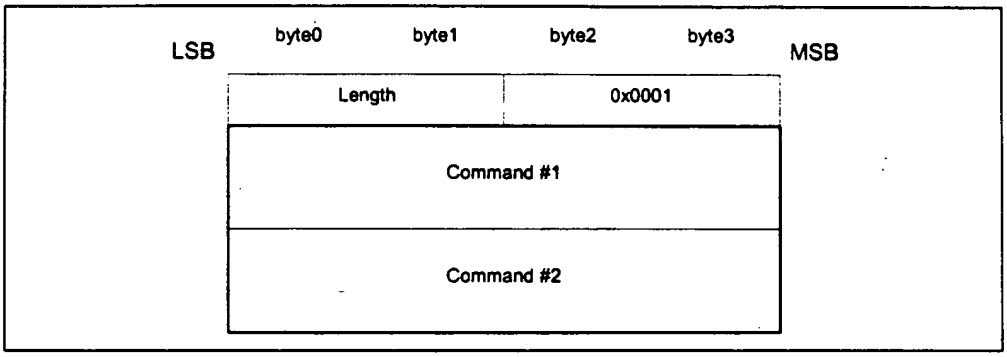


Figure 5.1: Signalling Command Packet Format

Figure 5.2 displays the general format of all signalling commands.

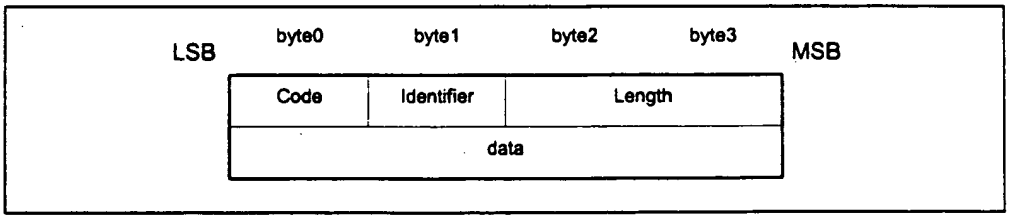


Figure 5.2: Command format

The fields shown are:

- *Code: 1 octet*

 The Code field is one octet long and identifies the type of command. When a packet is received with an unknown Code field, a Command Reject packet (defined in Section 5.1 on page 277) is sent in response.

 Up-to-date values of assigned Codes are specified in the latest Bluetooth 'Assigned Numbers' document (page 1009). Table 5.1 on page 276 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

Code	Description
0x00	RESERVED
0x01	Command reject
0x02	Connection request
0x03	Connection response
0x04	Configure request
0x05	Configure response
0x06	Disconnection request
0x07	Disconnection response
0x08	Echo request
0x09	Echo response
0x0A	Information request
0x0B	Information response

Table 5.1: Signalling Command Codes

- *Identifier: 1 octet*

 The Identifier field is one octet long and helps matching a request with the reply. The requesting device sets this field and the responding device uses the same value in its response. A different Identifier must be used for each original command. Identifiers should not be recycled until a period of 360 seconds has elapsed from the initial transmission of the command using the identifier. On the expiration of a RTX or ERTX timer, the same identifier should be used if a duplicate Request is re-sent as stated in Section 3.1.5 on page 262. A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded. Signalling identifier 0x0000 is defined to be an illegal identifier and shall never be used in any command.

- *Length: 2 octets*

 The Length field is two octets long and indicates the size in octets of the data field of the command only, i.e., it does not cover the Code, Identifier, and Length fields.

- *Data: 0 or more octets*

 The Data field is variable in length and discovered using the Length field. The Code field determines the format of the Data field.

5.1 COMMAND REJECT (CODE 0x01)

A Command Reject packet is sent in response to a command packet with an unknown command code or when sending the corresponding Response is inappropriate. Figure 5.3 displays the format of the packet. The Identifier should match the Identifier of the packet containing the unidentified code field. Implementations must always send these packets in response to unidentified signalling packets.

When multiple commands are included in an L2CAP packet and the packet exceeds the MTU of the receiver, a single Command Reject packet is sent in response. The identifier should match the first Request command in the L2CAP packet. If only Responses are recognized, the packet shall be silently discarded.

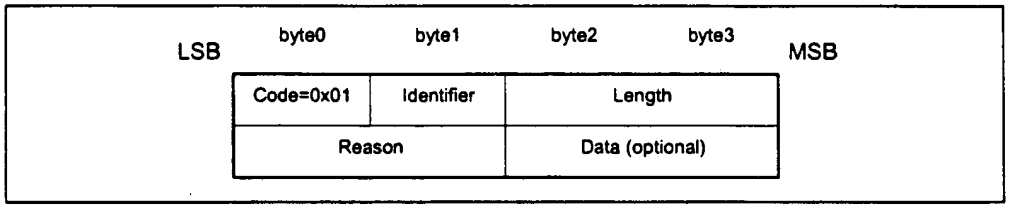


Figure 5.3: Command Reject Packet

- *Length = 0x0002 or more octets*
- *Reason: 2 octets*

 The Reason field describes why the Request packet was rejected.

Reason value	Description
0x0000	Command not understood
0x0001	Signalling MTU exceeded
0x0002	Invalid CID in request
Other	Reserved

Table 5.2: Reason Code Descriptions

- *Data: 0 or more octets*

 The length and content of the Data field depends on the Reason code. If the Reason code is 0x0000, "Command not understood", no Data field is used. If the Reason code is 0x0001, "Signalling MTU Exceeded", the 2-octet Data field represents the maximum signalling MTU the sender of this packet can accept.

 If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. A 4-octet data field on the command reject will contain the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The latter endpoints are obtained from the corresponding rejected command. If the rejected command contains only one of the channel endpoints, the other one is replaced by the null CID 0x0000.

Reason value	Data Length	Data value
0x0000	0 octets	N/A
0x0001	2 octets	Actual MTU
0x0002	4 octets	Requested CID

Table 5.3: Reason Data values

5.2 CONNECTION REQUEST (CODE 0x02)

Connection request packets are sent to create a channel between two devices. The channel connection must be established before configuration may begin. Figure 5.4 illustrates a Connection Request packet.

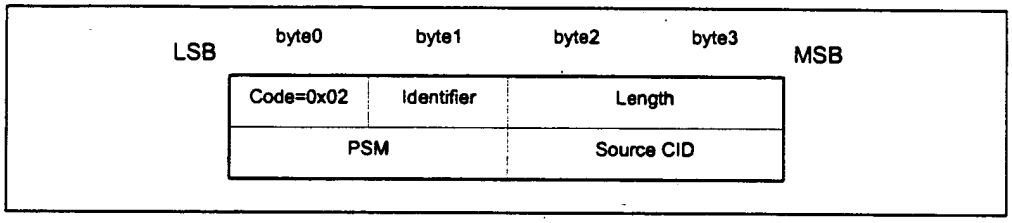


Figure 5.4: Connection Request Packet

- *Length = 0x0004 or more octets*
- *Protocol/Service Multiplexor (PSM): 2 octets (minimum)*

 The PSM field is two octets (minimum) in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values must be ODD, that is, the least significant bit of the least significant octet must be '1'. Also, all PSM values must be assigned such that the least significant bit of the most significant octet equals '0'. This allows the PSM field to be extended beyond 16 bits. PSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the Service Discovery Protocol (SDP). The dynamically assigned values may be used to support multiple implementations of a particular protocol, e.g., RFCOMM, residing on top of L2CAP or for prototyping an experimental protocol.

PSM value	Description
0x0001	Service Discovery Protocol
0x0003	RFCOMM
0x0005	Telephony Control Protocol
<0x1000	RESERVED
[0x1001-0xFFFF]	DYNAMICALLY ASSIGNED

Table 5.4: Defined PSM Values

- *Source CID (SCID): 2 octets*

 The source local CID is two octets in length and represents a channel end-point on the device sending the request. Once the channel has been configured, data packets flowing from the sender of the request must be send to this CID. In this section, the Source CID represents the channel endpoint on the device sending the request and receiving the response, while the Destination CID represents the channel endpoint on the device receiving the request and sending the response.

5.3 CONNECTION RESPONSE (CODE 0x03)

When a unit receives a Connection Request packet, it must send a Connection Response packet. The format of the connection response packet is shown in Figure 5.5.

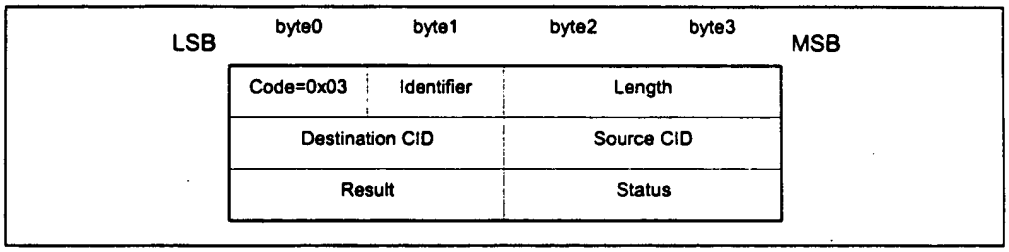


Figure 5.5: Connection Response Packet

- *Length = 0x0008 octets*
- *Destination Channel Identifier (DCID): 2 octets*

 The field contains the channel end-point on the device sending this Response packet.
- *Source Channel Identifier (SCID): 2 octets*

 The field contains the channel end-point on the device receiving this Response packet.

- *Result: 2 octets*

 The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed. A logical channel is established on the receipt of a successful result. Table 5.5 defines values for this field. If the result field is not zero, the DCID and SCID fields should be ignored.

Value	Description
0x0000	Connection successful.
0x0001	Connection pending
0x0002	Connection refused – PSM not supported.
0x0003	Connection refused – security block.
0x0004	Connection refused – no resources available.
Other	Reserved.

Table 5.5: Result values

- *Status: 2 octets*

 Only defined for Result = Pending. Indicates the status of the connection.

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorization pending
Other	Reserved

Table 5.6: Status values

5.4 CONFIGURATION REQUEST (CODE 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to re-negotiate this contract whenever appropriate. During a re-negotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request is related exclusively either with the outgoing or the incoming data traffic but not both of them. In Section 6 on page 289, the various configuration parameters and their relation to the outgoing or incoming data traffic are presented. If an L2CAP entity receives a Configuration Request while it is waiting for a response it must not block sending the Configuration Response, otherwise the configuration process may deadlock.

If no parameters need to be negotiated, no options need to be inserted and the C-bit should be cleared. L2CAP entities in remote devices MUST negotiate all parameters defined in this document whenever the default values are not

acceptable. Any missing configuration parameters are assumed to have their most recently (mutually) explicitly or implicitly accepted values. Event if all default values are acceptable, a Configuration Request packet with no options MUST be sent. Implicitly accepted values are any default values for the configuration parameters specified in this document that have not been explicitly negotiated for the specific channel under configuration.

Each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. If a device needs to establish the value of a configuration parameter in the opposite direction than the one implied by a Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Connection Request.

The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation but it shall not last more than 120 seconds.

Figure 5.6 defines the format of the Configuration Request packet.

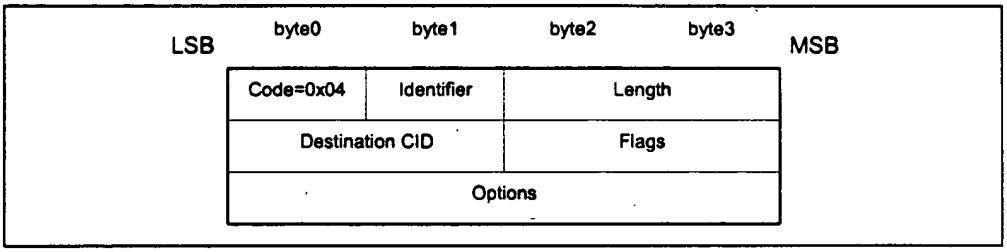


Figure 5.6: Configuration Request Packet

- *Length = 0x0004 or more octets*
- *Destination CID (DCID): 2 octets*

 The field contains the channel end-point on the device receiving this Request packet.
- *Flags: 2 octets*

 Figure 5.7 display the two-octet Flags field. Note the most significant bit is shown on the left.

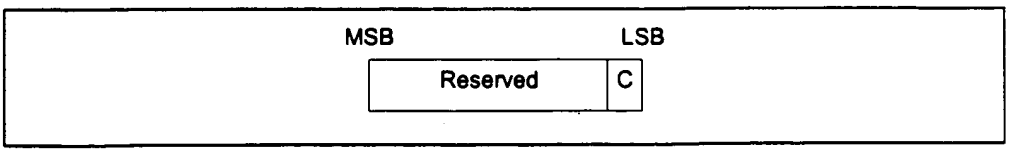


Figure 5.7: Configuration Request Flags field format

C – more configuration requests will follow when set to 1. This flag indicates that the remote device should not enter OPEN state after agreeing to these parameters because more parameter negotiations are being sent. Segment-

ing the Configuration Request packet is necessary if the parameters exceed the MTU_{sig}.

Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

- *Configuration Options*

 The list of the parameters and their values to be negotiated. These are defined in Section 6 on page 289. Configuration Requests may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the length field is set to 0x0004.

0656565 0440409

5.5 CONFIGURE RESPONSE (CODE 0X05)

Configure Response packets MUST be sent in reply to Configuration Request packets. Each configuration parameter value (if any is present) in a Configuration Response reflects an 'adjustment' to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request. Thus, for example, if a configuration parameter in a Configuration Request relates to traffic flowing from device A to device B, the sender of the Configuration Response will only adjust (if needed) this value again for the same traffic flowing from device A to device B. The options sent in the Response depend on the value in the Result field. Figure 5.8 defines the format of the Configuration Response packet.

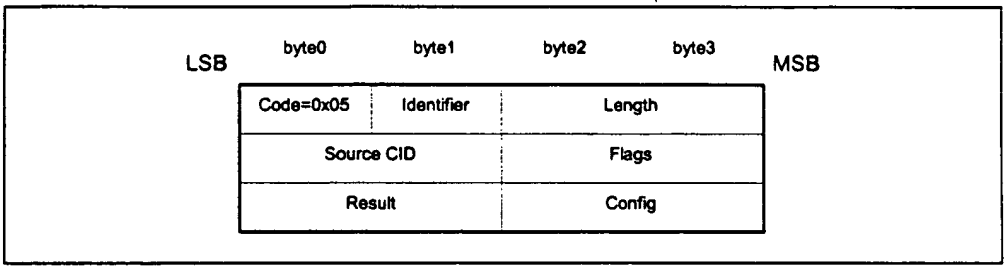


Figure 5.8: Configuration Response Packet

- *Length = 0x0006 or more octets*
- *Source CID (SCID): 2 octets*

 The field contains the channel end-point on the device receiving this Response packet. The device receiving the Response must check that the Identifier field matches the same field in the corresponding configuration request command and the SCID matches its local CID paired with the original DCID.
- *Flags: 2 octets*

 Figure 5.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

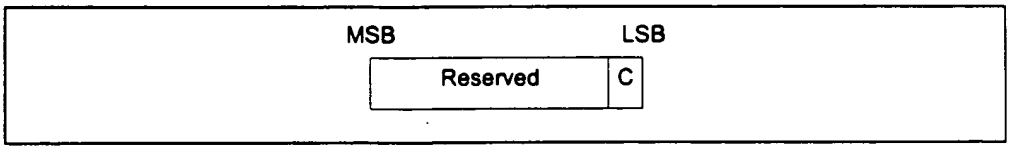


Figure 5.9: Configuration Response Flags field format

C – more configuration responses will follow when set to 1. This flag indicates that the parameters included in the response are a partial subset of parameters being sent by the device sending the Response packet.

Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

- *Result: 2 octets*

 The Result field indicates whether or not the Request was acceptable. See Table 5.7 for possible result codes.

Result	Description
0x0000	Success
0x0001	Failure – unacceptable parameters
0x0002	Failure – rejected (no reason provided)
0x0003	Failure – unknown options
Other	RESERVED

Table 5.7: Configuration Response Result codes

- *Configuration Options*

 This field contains the list of parameters being negotiated. These are defined in Section 6 on page 289. On a successful result, these parameters contain the return values for any wild card parameters (see Section 6.3 on page 291) contained in the request.

 On an unacceptable parameters failure (Result=0x0001) the rejected parameters should be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters are assumed to have their most recently (mutually) accepted values and they too can be included in the Configuration Response if need to be changed. Recall that, each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. Thus, if the sender of the Configuration Response needs to establish the value of a configuration parameter in the opposite direction than the one implied by an original Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Connection Request.

 On an unknown option failure (Result=0x0003), the option types not understood by the recipient of the Request must be included in the Response. Note that hints (defined in Section 6 on page 289), those options in the Request that are skipped if not understood, must not be included in the Response and must not be the sole cause for rejecting the Request.

 The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation.

5.6 DISCONNECTION REQUEST (CODE 0x06)

Terminating an L2CAP channel requires that a disconnection request packet be sent and acknowledged by a disconnection response packet. Disconnection is requested using the signalling channel since all other L2CAP packets sent to the destination channel automatically get passed up to the next protocol layer. Figure 5.10 displays a disconnection packet request. The receiver must ensure both source and destination CIDs match before initiating a connection disconnection. Once a Disconnection Request is issued, all incoming data in transit on this L2CAP channel will be discarded and any new additional outgoing data is not allowed. Once a disconnection request for a channel has been received, all data queued to be sent out on that channel may be discarded.

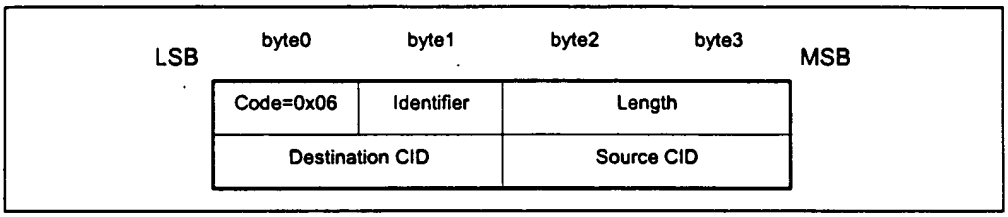


Figure 5.10: Disconnection Request Packet

- *Length = 0x0004 octets*
- *Destination CID (DCID): 2 octets*

 This field specifies the end-point of the channel to be shutdown on the device receiving this request.
- *Source CID (SCID): 2 octets*

 This field specifies the end-point of the channel to be shutdown on the device sending this request.

 The SCID and DCID are relative to the sender of this request and must match those of the channel to be disconnected. If the DCID is not recognized by the receiver of this message, a CommandReject message with 'invalid CID' result code must be sent in response. If the receivers finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.

5.7 DISCONNECTION RESPONSE (CODE 0x07)

Disconnection responses should be sent in response to each disconnection request.

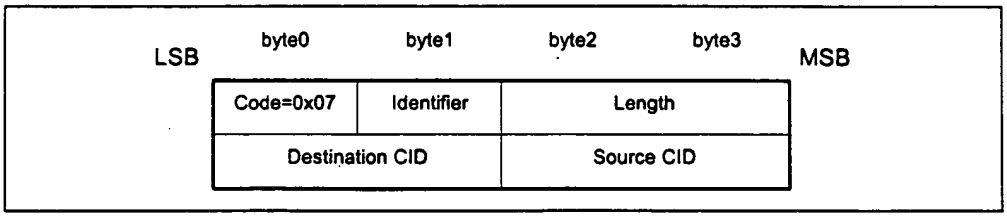


Figure 5.11: Disconnection Response Packet

- *Length = 0x0004 octets*
- *Destination CID (DCID): 2 octets*

 This field identifies the channel end-point on the device sending the response.
- *Source CID (SCID): 2 octets*

 This field identifies the channel end-point on the device receiving the response.

 The DCID and the SCID (which are relative to the sender of the request), and the Identifier fields must match those of the corresponding disconnection request command. If the CIDs do not match, the response should be silently discarded at the receiver.

5.8 ECHO REQUEST (CODE 0x08)

Echo requests are used to solicit a response from a remote L2CAP entity. These requests may be used for testing the link or passing vendor specific information using the optional data field. L2CAP entities MUST respond to well-formed Echo Request packets with an Echo Response packet. The Data field is optional and implementation-dependent. L2CAP entities should ignore the contents of this field.

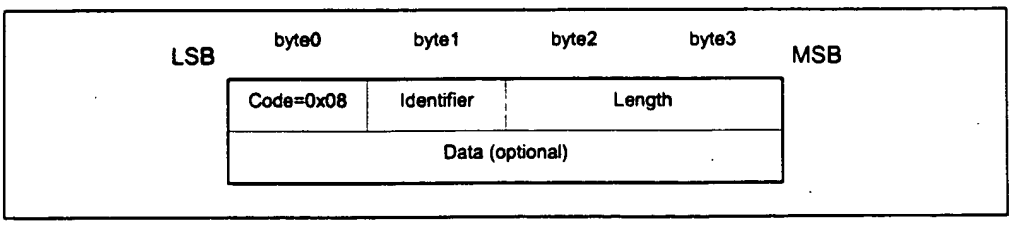


Figure 5.12: Echo Request Packet

5.9 ECHO RESPONSE (CODE 0x09)

Echo responses are sent upon receiving Echo Request packets. The identifier in the response MUST match the identifier sent in the Request. The optional and implementation-dependent data field may contain the contents of the data field in the Request, different data, or no data at all.

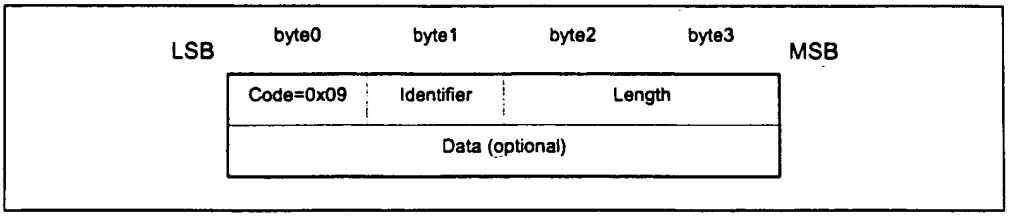


Figure 5.13: Echo Response Packet

5.10 INFORMATION REQUEST (CODE 0X0A)

Information requests are used to solicit implementation-specific information from a remote L2CAP entity. L2CAP entities MUST respond to well-formed Information Request packets with an Information Response packet.

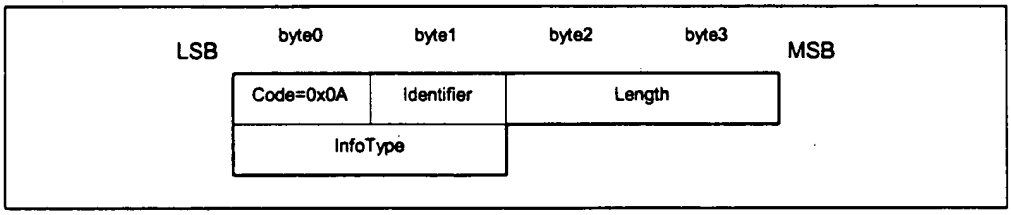


Figure 5.14: Information Request Packet

- *Length = 0x0002 octets*
- *InfoType: 2 octets*

 The InfoType defines the type of implementation-specific information being solicited.

Value	Description
0x0001	Connectionless MTU
Other	Reserved

Table 5.8: InfoType definitions

5.11 INFORMATION RESPONSE (CODE 0X0B)

Information responses are sent upon receiving Information Request packets. The identifier in the response MUST match the identifier sent in the Request. The optional data field may contain the contents of the data field in the Request, different data, or no data at all.

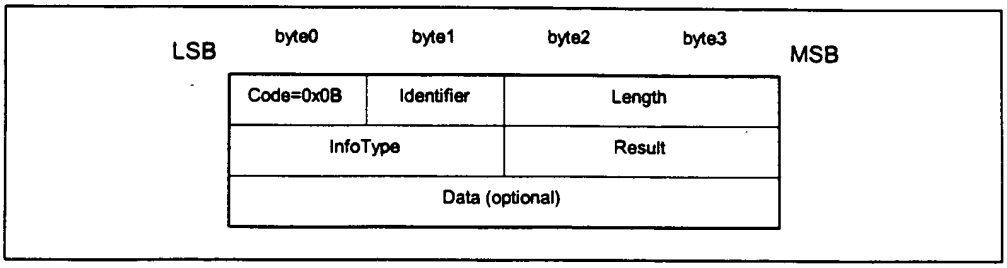


Figure 5.15: Information Response Packet

- *InfoType: 2 octets*

 Same value sent in the request.

- *Result: 2 octets*

 The Result contains information about the success of the request. If result is "Success", the data field contains the information as specified in Table 5.10. If result is "Not supported", no data should be returned.

Value	Description
0x0000	Success
0x0001	Not supported
Other	Reserved

Table 5.9: Information Response Result values

- *Data: 0 or more octets*

 The contents of the Data field depends on the InfoType. For the Connection MTU request, the data field contains the remote entity's 2-octet acceptable connectionless MTU.

InfoType	Data	Data Length (in octets)
0x0001	Connectionless MTU	2

Table 5.10: Information Response Data fields

6 CONFIGURATION PARAMETER OPTIONS

Options are a mechanism to extend the ability to negotiate different connection requirements. Options are transmitted in the form of information elements comprised an option type, an option length, and one or more option data fields. Figure 6.1 illustrates the format of an option.

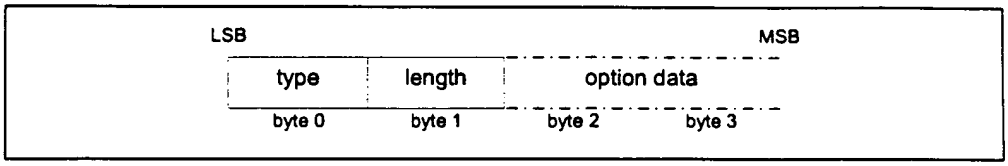


Figure 6.1: Configuration option format

- *Type: 1 octet*

 The option type field defines the parameters being configured. The most significant bit of the type determines the action taken if the option is not recognized. The semantics assigned to the bit are defined below.

 0 - option must be recognized; refuse the configuration request

 1 - option is a hint; skip the option and continue processing

- *Length: 1 octet*

 The length field defines the number of octets in the option payload. So an option type with no payload has a length of 0.

- *Option data*

 The contents of this field are dependent on the option type.

6.1 MAXIMUM TRANSMISSION UNIT (MTU)

This option specifies the payload size the sender is capable of accepting. The type is 0x01, and the payload length is 2 bytes, carrying the two-octet MTU size value as the only information element (see Figure 6.2 on page 290).

Since all L2CAP implementations are capable to support a minimum L2CAP packet size, see Section 4 on page 272, MTU is not really a negotiated value but rather an informational parameter to the remote device that the local device can accommodate in this channel an MTU larger than the minimum required. In the unlikely case that the remote device is only willing to send L2CAP packets in this channel that are larger than the MTU announced by the local device, then this Configuration Request will receive a negative response in which the remote device will include the value of MTU that is indented to transmit. In this case, it is implementation specific on whether the local device will continue the configuration process or even maintain this channel.

The remote device in its positive Configuration Response will include the actual MTU to be used on this channel for traffic flowing into the local device which is

minimum{ MTU in configReq, outgoing MTU capability of remote device }. The MTU to be used on this channel but for the traffic flowing in the opposite direction will be established when the remote device (with respect to this discussion) sends its own Configuration Request as explained in Section 5.4 on page 280.

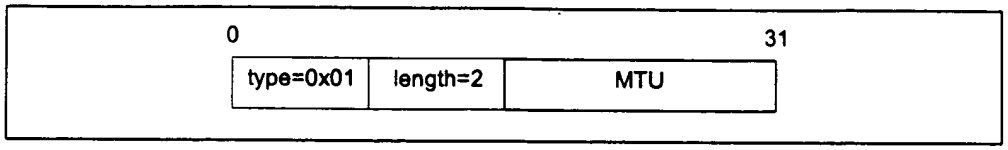


Figure 6.2: MTU Option Format

- Maximum Transmission Unit (MTU) Size: 2 octets

 The MTU field represents the largest L2CAP packet payload, in bytes, that the originator of the Request can accept for that channel. The MTU is asymmetric and the sender of the Request shall specify the MTU it can receive on this channel if it differs from the default value. L2CAP implementations must support a minimum MTU size of 48 bytes. The default value is 672 bytes[1].

6.2 FLUSH TIMEOUT OPTION

This option is used to inform the recipient of the amount of time the originator's link controller / link manager will attempt to successfully transmit an L2CAP segment before giving up and flushing the packet. The type is 0x02 and the payload size is 2 octets.

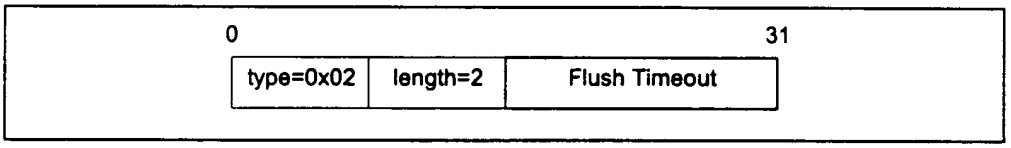


Figure 6.3: Flush Timeout

- *Flush Timeout*

 This value represents units of time measured in milliseconds. The value of 1 implies no retransmissions at the Baseband level should be performed since the minimum polling interval is 1.25 ms. The value of all 1's indicates an infinite amount of retransmissions. This is also referred to as 'reliable channel'. In this case, the link manager shall continue retransmitting a segment until physical link loss occurs. This is an asymmetric value and the sender of the Request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

1. The default MTU was selected based on the payload carried by two Baseband DH5 packets (2*341=682) minus the Baseband ACL headers (2*2=4) and L2CAP header (6).

6.3 QUALITY OF SERVICE (QOS) OPTION

This option specifies a flow specification (flowSpec) similar to RFC 1363 [1]. If no QoS configuration parameter is negotiated the link should assume the default parameters discussed below. The QoS option is type 0x03.

When included in a Configuration Request, this option describes the outgoing traffic flow from the device sending the request to the device receiving it. When included in a positive Configuration Response, this option describes the incoming traffic flow agreement as seen from the device sending the response. When included in a negative Configuration Response, this option describes the preferred incoming traffic flow from the perspective of the device sending the response.

L2CAP implementations are only required to support 'Best Effort' service, support for any other service type is optional. Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort must be assumed. If any QoS guarantees are required then a QoS configuration request must be sent.

The remote device places information that depends on the value of the result field, see Section 5.5 on page 283, in its Configuration Response. If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is "Failure – unacceptable parameters", the response may include a list of outgoing flowspec parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response. Recall that any missing configuration parameters from a Configuration Request are assumed to have their most recently (mutually) accepted values. For both Best effort and Guaranteed service, when the QoS option appears in the Configuration Response, "do not cares" shall be present where they appeared in the Configuration Request.

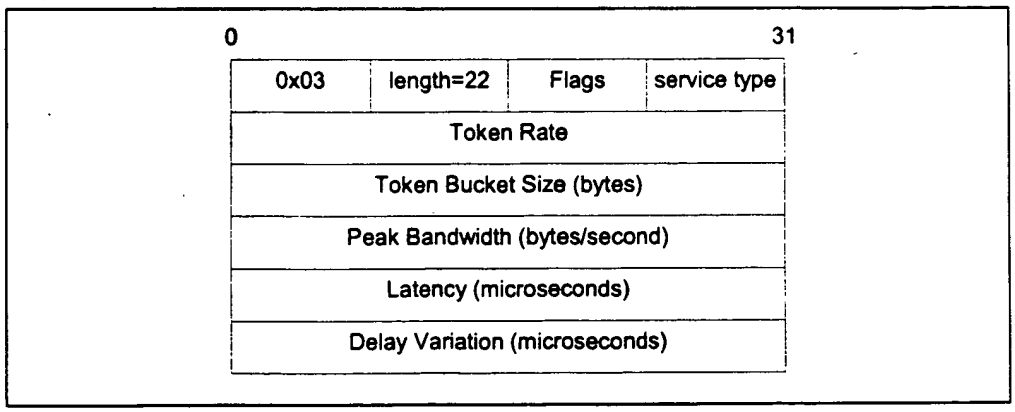


Figure 6.4: Quality of Service Flow Specification

- *Flags: 1 octet*

 Reserved for future use and must be set to 0.

- *Service Type: 1 octet*

 This field indicates the level of service required. Table 6.1 defines the different services available. If 'No traffic' is selected, the remainder of the fields may be ignored because there is no data being sent across the channel in the outgoing direction.

 If 'Best effort', the default value, is selected, the remaining fields should be treated as hints by the remote device. The remote device may choose to ignore the fields, try to satisfy the hint but provide no response (QoS option omitted in the Response message), or respond with the settings it will try to meet.

Value	Description
0x00	No traffic
0x01	Best effort (Default)
0x02	Guaranteed
Other	Reserved

Table 6.1: Service type definitions

- *Token Rate: 4 octets*

 The value of this field represents the rate at which traffic credits are granted in bytes per second. An application may send data at this rate continuously. Burst data may be sent up to the token bucket size (see below). Until that data burst has been drained, an application must limit itself to the token rate. The value 0x00000000 indicates no token rate is specified. This is the default value and implies indifference to token rate. The value 0xFFFFFFFF represents a wild card matching the maximum token rate available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

- *Token Bucket Size: 4 octets*

 The value of this field represents the size of the token bucket in bytes. If the bucket is full, then applications must either wait or discard data. The value of 0x00000000 represents no token bucket is needed; this is the default value. The value 0xFFFFFFFF represents a wild card matching the maximum token bucket available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum buffer space available at the time of the request.

- *Peak Bandwidth: 4 octets*

The value of this field, expressed in bytes per second, limits how fast packets may be sent back-to-back from applications. Some intermediate systems can take advantage of this information, resulting in more efficient resource allocation. The value of 0x00000000 states that the maximum bandwidth is unknown, which is the default value.

- *Latency: 4 octets*

 The value of this field represents the maximum acceptable delay between transmission of a bit by the sender and its initial transmission over the air, expressed in microseconds. The precise interpretation of this number depends on the level of guarantee specified in the Class of Service. The value 0xFFFFFFFF represents a do not care and is the default value.

- *Delay Variation: 4 octets*

 The value of this field is the difference, in microseconds, between the maximum and minimum possible delay that a packet will experience. This value is used by applications to determine the amount of buffer space needed at the receiving side in order to restore the original data transmission pattern. The value 0xFFFFFFFF represents a do not care and is the default value.

6.4 CONFIGURATION PROCESS

Negotiating the channel parameters involves three steps:

1. Informing the remote side of the non-default parameters that the local side will accept
2. Having the remote side agreeing or disagreeing to these values (including the default ones); steps (1) and (2) may iterate as needed
3. Repeat steps (1) and (2) for the reverse direction from the (previous) remote side to the (previous) local side.

This process can be abstracted into a Request negotiation path and a Response negotiation path.

6.4.1 Request Path

The Request Path negotiates the incoming MTU, flush timeout, and outgoing flowspec. Table 6.2 defines the configuration options that may be placed in the Configuration Request message and their semantics.

Parameter	Description
MTU	Incoming MTU information
FlushTO	Outgoing flush timeout
OutFlow	Outgoing flow information.

Table 6.2: Parameters allowed in Request

6.4.2 Response Path

The Response Path negotiates the outgoing MTU (remote side's incoming MTU), the remote side's flush timeout, and incoming flowspec (remote side's outgoing flowspec). If a request-oriented parameter is not present in the Request message (reverts to default value), the remote side may negotiate for a non-default value by including the proposed value in a negative Response message.

Parameter	Description
MTU	Outgoing MTU information
FlushTO	Incoming flush timeout
InFlow	Incoming flow information

Table 6.3: Parameters allowed in Response

6.4.3 Configuration State Machine

The configuration state machine shown below depicts two paths. Before leaving the CONFIG state and moving into the OPEN state, both paths must reach closure. The request path requires the local device to receive a positive response to reach closure while the response path requires the local device to send a positive response to reach closure.

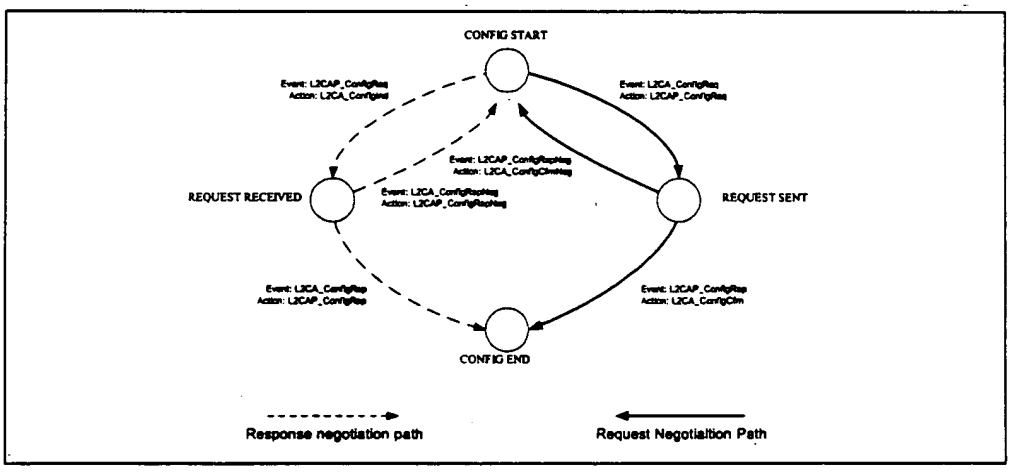


Figure 6.5: Configuration State Machine

"Appendix A: Configuration MSCs" on page 318 provides some configuration examples.

7 SERVICE PRIMITIVES

This section presents an abstract description of the services offered by L2CAP in terms of service primitives and parameters. The service interface is required for testing. The interface is described independently of any platform specific implementation. All data values use Little Endian byte ordering.

7.1 EVENT INDICATION

Service	Input Parameters	Output Parameters
EventIndication	Event, Callback	Result

Description:

The use of this primitive requests a callback when the selected indication Event occurs.

Input Parameters:

Event *Type: uint* *Size: 2 octets*

Value	Description
0x00	Reserved
0x01	L2CA_ConnectInd
0x02	L2CA_ConfigInd
0x03	L2CA_DisconnectInd
0x04	L2CA_QoSViolationInd
other	Reserved for future use

Callback *Type: function* *Size: N/A*

Event	Callback Function Input Parameters
L2CA_ConnectInd	BD_ADDR, CID, PSM, Identifier
L2CA_ConfigInd	CID, OutMTU, InFlow, InFlushTO
L2CA_DisconnectInd	CID
L2CA_QoSViolationInd	BD_ADDR

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Event successfully registered
0x0001	Event registration failed

7.1.1 L2CA_ConnectInd Callback

This callback function includes the parameters for the address of the remote device that issued the connection request, the local CID representing the channel being requested, the Identifier contained in the request, and the PSM value the request is targeting.

7.1.2 L2CA_ConfigInd Callback

This callback function includes the parameters indicating the local CID of the channel the request has been sent to, the outgoing MTU size (maximum packet that can be sent across the channel) and the flowspec describing the characteristics of the incoming data. All other channel parameters are set to their default values if not provided by the remote device.

7.1.3 L2CA_DisconnectInd Callback

This callback function includes the parameter indicating the local CID the request has been sent to.

7.1.4 L2CA_QoSViolationInd Callback

This callback function includes the parameter indicating the address of the remote Bluetooth device where the QoS contract has been violated.

7.2 CONNECT

Service	Input Parameters	Output Parameters
L2CA_ConnectReq	PSM, BD_ADDR	LCID, Result, Status

Description:

This primitive initiates the sending of an L2CA_ConnectReq message and blocks until a corresponding L2CA_ConnectCfm(Neg) or L2CA_TimeOutInd message is received.

The use of this primitive requests the creation of a channel representing a logical connection to a physical address. Input parameters are the target protocol (*PSM*) and remote device's 48-bit address (*BD_ADDR*). Output parameters are the local CID *(LCID)* allocated by the local L2CAP entity, and *Result* of the request. If the *Result* indicates success, the *LCID* value contains the identification of the local endpoint. Otherwise the *LCID* returned should be set to 0. If *Result* indicates a pending notification, the *Status* value may contain more information of what processing is delaying the establishment of the connection. Otherwise the *Status* value should be ignored.

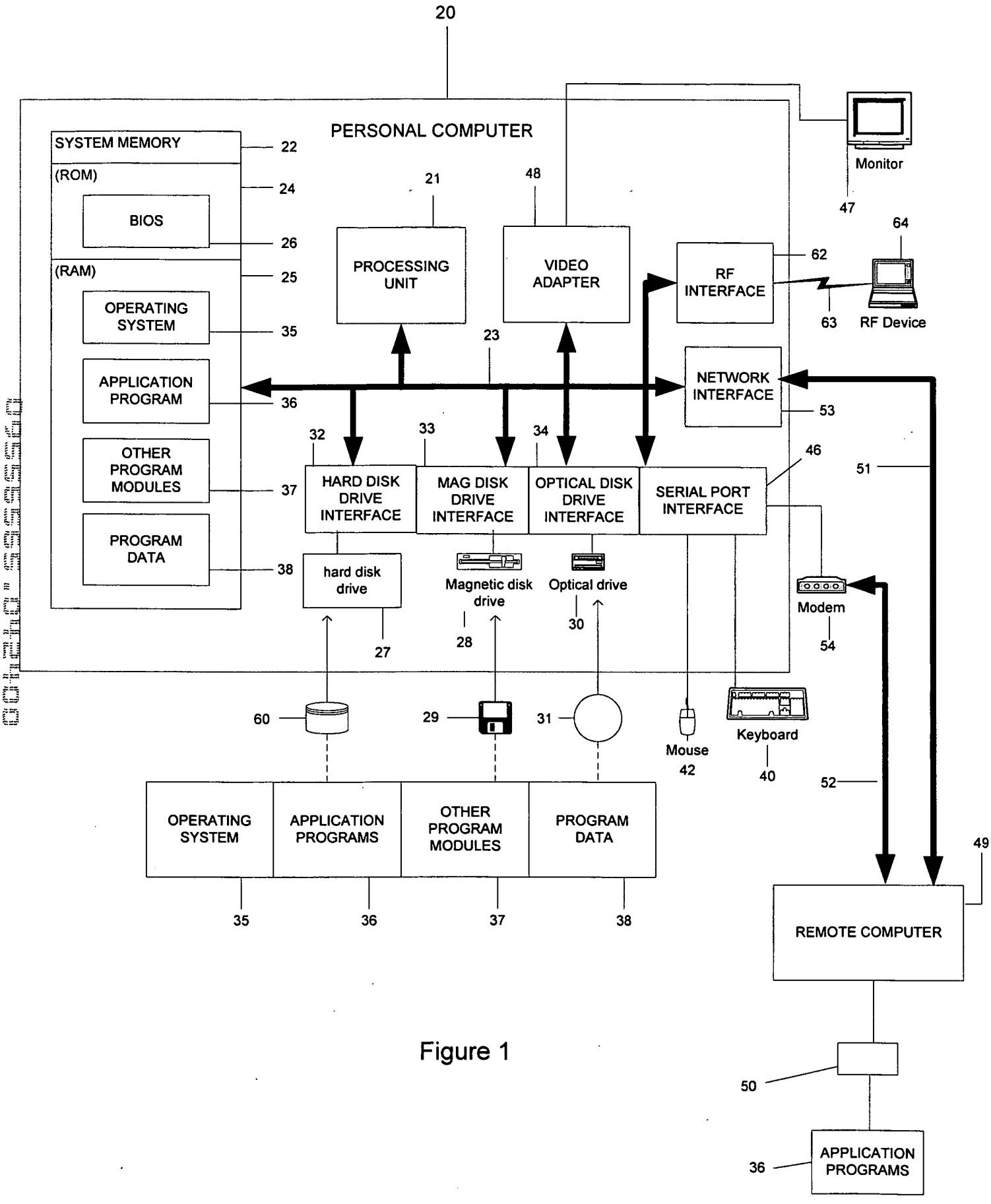

20
PERSONAL COMPUTER
SYSTEM MEMORY
22
(ROM)
24
BIOS
26
(RAM)
25
OPERATING SYSTEM
35
APPLICATION PROGRAM
36
OTHER PROGRAM MODULES
37
PROGRAM DATA
38
21
PROCESSING UNIT
48
VIDEO ADAPTER
23
Monitor
47
64
RF INTERFACE
62
63
RF Device
NETWORK INTERFACE
53
32
33
34
46
51
HARD DISK DRIVE INTERFACE
MAG DISK DRIVE INTERFACE
OPTICAL DISK DRIVE INTERFACE
SERIAL PORT INTERFACE
hard disk drive
27
Magnetic disk drive
28
Optical drive
30
Modem
54
60
29
31
Mouse
42
Keyboard
40
52
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
35
36
37
38
49
REMOTE COMPUTER
50
APPLICATION PROGRAMS
36

Figure 1

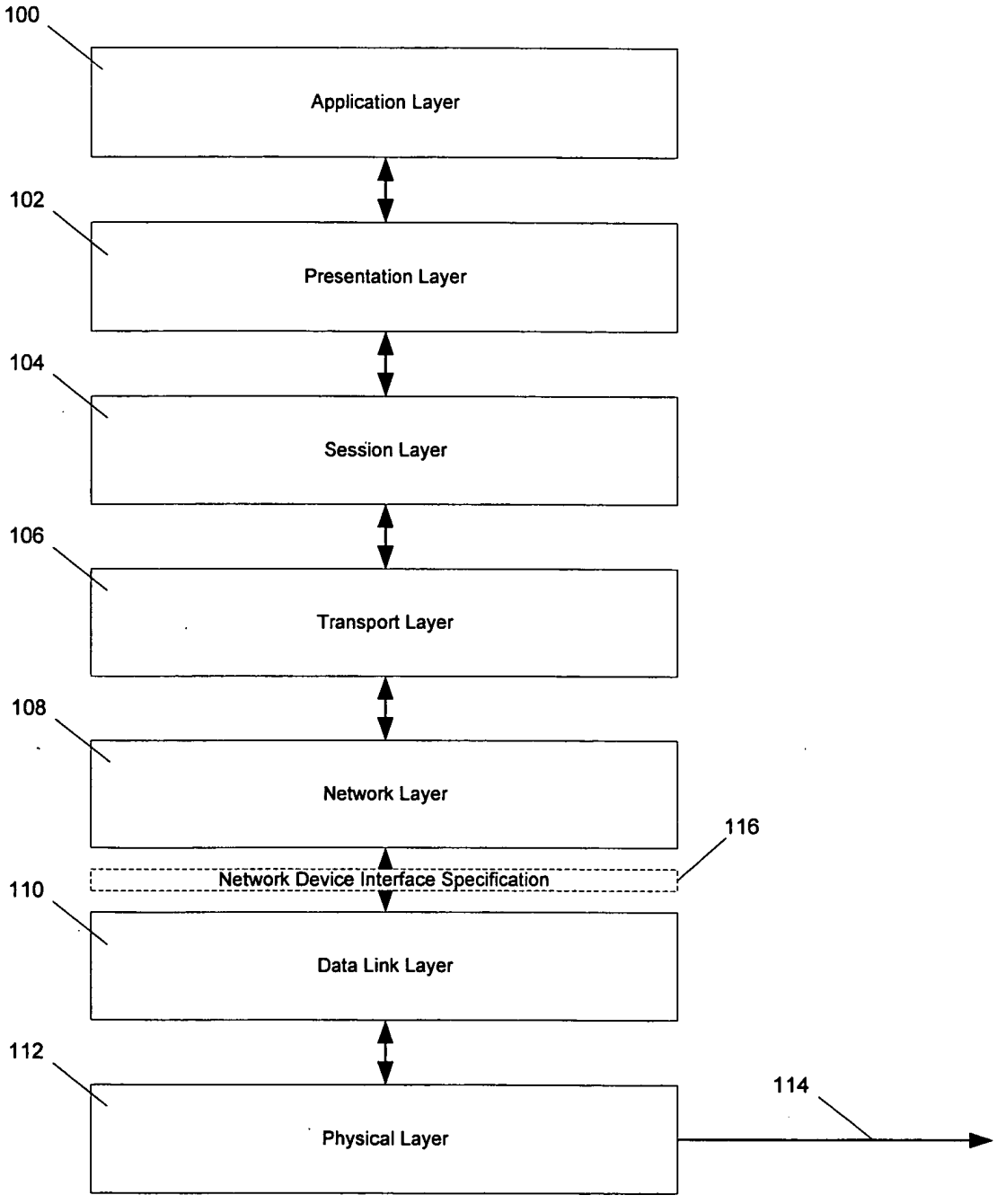

100
Application Layer
102
Presentation Layer
104
Session Layer
106
Transport Layer
108
Network Layer
116
Network Device Interface Specification
110
Data Link Layer
112
Physical Layer
114

Figure 2

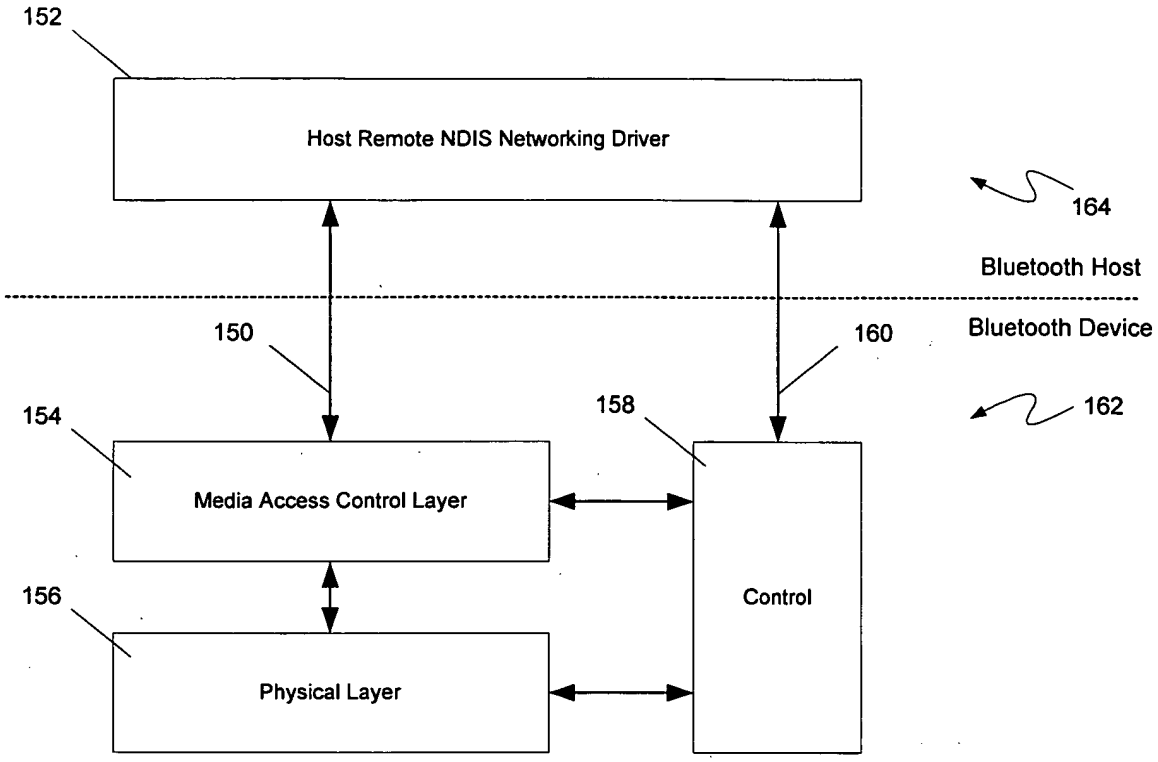

152
Host Remote NDIS Networking Driver
164
Bluetooth Host
Bluetooth Device
150
160
158
162
154
Media Access Control Layer
156
Control
Physical Layer

Figure 3

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Target PSM provided for the connection

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Output Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel if Result = 0x0000, otherwise set to 0.

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful and the CID identifies the local endpoint. Ignore Status parameter
0x0001	Connection pending. Check Status parameter for more information
0x0002	Connection refused because no service for the PSM has been registered
0x0003	Connection refused because the security architecture on the remote side has denied the request
0xEEEE	Connection timeout occurred. This is a result of a timer expiration indication being included in the connection confirm message

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information
0x0001	Authentication pending
0x0002	Authorization pending

7.3 CONNECT RESPONSE

Service	Input Parameters	Output Parameters
L2CA_ConnectRsp	BD_ADDR, Identifier, LCID, Response, Status	Result

Description:

This primitive represents the L2CA_ConnectRsp.

The use of this primitive issues a response to a connection request event indication. Input parameters are the remote device's 48-bit address, Identifier sent in the request, local CID, the Response code, and the Status attached to the Response code. The output parameter is the Result of the service request.

This primitive must be called no more than once after receiving the callback indication. This primitive returns once the local L2CAP entity has validated the request. A successful return does indicate the response has been sent over the air interface.

Input Parameters:

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Identifier *Type: uint* *Size: 1 octets*

Value	Description
0xXX.	This value must match the value received in the L2CA_ConnectInd event described in Section 7.1.1 on page 296

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Response *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful
0x0001	Connection pending
0x0002	Connection refused – PSM not supported
0x0003	Connection refused – security block
0x0004	Connection refused – no resources available
0xXXXX	Other connection response code

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorization pending
0xXXXX	Other status code

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response successfully sent
0x0001	Failure to match any outstanding connection request

7.4 CONFIGURE

Service	Input Parameters	Output Parameters
L2CA_ConfigReq	CID, InMTU, OutFlow, OutFlushTO, LinkTO	Result, InMTU, OutFlow, OutFlushTO

Description:

This primitive initiates the sending of an L2CA_ConfigReq message and blocks until a corresponding L2CA_ConfigCfm(Neg) or L2CA_TimeOutInd message is received.

The use of this primitive requests the initial configuration (or reconfiguration) of a channel to a new set of channel parameters. Input parameters are the local CID endpoint, new incoming receivable MTU (InMTU), new outgoing flow specification, and flush and link timeouts. Output parameters composing the L2CA_ConfigCfm(Neg) message are the Result, accepted incoming MTU(InMTU), the remote side's flow requests, and flush and link timeouts. Note that the output results are returned only after the local L2CAP entity transitions out of the CONFIG state (even if this transition is back to the CONFIG state).

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Local CID

InMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel can accept

OutFlow *Type: Flow* *Size: x octets*

Value	Description
flowspec	Quality of service parameters dealing with the traffic characteristics of the outgoing data flow

OutFlushTO *Size 2 octets*

Value	Description
0xXXXX	Number of milliseconds to wait before an L2CAP packet that cannot be acknowledged at the physical layer is dropped
0x0000	Request to use the existing flush timeout value if one exists, otherwise the default value (0xFFFF) will be used
0x0001	Perform no retransmissions at the Baseband layer
0xFFFF	Perform retransmission at the Baseband layer until the link timeout terminates the channel

LinkTO *Size 2 octets*

Value	Description
0xXXXX	Number of milliseconds to wait before terminating an unresponsive link

Output Parameters:

Result *Size 2 octets*

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values
0x0001	Failure – invalid CID
0x0002	Failure – unacceptable parameters
0x0003	Failure – signalling MTU exceeded
0x0004	Failure – unknown options
0xEEEE	Configuration timeout occurred. This is a result of a timer expiration indication being included in the configuration confirm

InMTU *Size 2 octets*

Value	Description
0xXXXX	Maximum transmission unit that the remote unit will send across this channel (maybe less or equal to the InMTU input parameter).

OutFlow *Size 2 octets*

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the agreed-upon outgoing data flow if Result is successful. Otherwise this represents the requested Quality of Service

OutFlushTO *Size 2 octets*

Value	Description
0xXXXX	Number of milliseconds before an L2CAP packet that cannot be acknowledged at the physical layer is dropped. This value is informative of the actual value that will be used for outgoing packets. It may be less or equal to the OutFlushTO parameter given as input.

7.5 CONFIGURATION RESPONSE

Service	Input Parameters	Output Parameters
L2CA_ConfigRsp	CID, OutMTU, InFlow	Result

Description:

This primitive represents the L2CAP_ConfigRsp.

The use of this primitive issues a response to a configuration request event indication. Input parameters include the local CID of the endpoint being configured, outgoing transmit MTU (which may be equal or less to the OutMTU parameter in the L2CA_ConfigInd event) and the accepted flowspec for incoming traffic. The output parameter is the Result value.

Input Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Local channel identifier

OutMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel will send

InFlow *Type: Flow* *Size: x octets*

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the incoming data flow

Output Parameters:

Result *Size 2 octets*

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values
0x0001	Configuration failed – unacceptable parameters
0x0002	Configuration failed – rejected
0x0003	Configuration failed – invalid CID
0x0004	Configuration failed – unknown options
0xXXXX	Reserved

7.6 DISCONNECT

Service	Input Parameters	Output Parameters
L2CA_DisconnectReq	CID	Result

Description:

This primitive represents the L2CAP_DisconnectReq and the returned output parameters represent the corresponding L2CAP_DisconnectRsp or the RTX timer expiration.

The use of this primitive requests the disconnection of the channel. Input parameter is the *CID* representing the local channel endpoint. Output parameter is *Result*. *Result* is zero if a L2CAP_DisconnectRsp is received, otherwise a non-zero value is returned. Once disconnection has been requested, no process will be able to successfully read or write from the CID. Writes in progress should continue to be processed.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Disconnection successful. This is a result of the receipt of a disconnection response message
0xEEEE	Disconnection timeout occurred.

7.7 WRITE

Service	Input Parameters	Output Parameters
L2CA_DataWrite	CID, Length, OutBuffer	Size, Result

Description:

The use of this primitive requests the transfer of data across the channel. If the length of the data exceeds the OutMTU then only the first OutMTU bytes are sent This command may be used for both connection-oriented and connection-less traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where data to be transmitted are stored

OutBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the input buffer used to store the message

Output Parameters:

Size *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	The number of bytes transferred

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful write
0x0001	Error – Flush timeout expired
0x0002	Error – Link termination (perhaps this should be left to the indication)

7.8 READ

Service	Input Parameters	Output Parameters
L2CA_DataRead	CID, Length, InBuffer	Result, N

Description:

The use of this primitive requests for the reception of data. This request returns when data is available or the link is terminated. The data returned represents a single L2CAP payload. If not enough data is available, the command will block until the data arrives or the link is terminated. If the payload is bigger than the buffer, only the portion of the payload that fits into the buffer will be returned, and the remainder of the payload will be discarded. This command may be used for both connection-oriented and connectionless traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	CID

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where received data are to be stored

InBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the buffer used to store the message

Output parameters:

Result

Value	Description
0x0000	Success

N *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Number of bytes transferred to InBuffer

7.9 GROUP CREATE

Service	Input Parameters	Output Parameters
L2CA_GroupCreate	PSM	CID

Description:

The use of this primitive requests the creation of a CID to represent a logical connection to multiple devices. Input parameter is the *PSM* value that the outgoing connectionless traffic is labelled with, and the filter used for incoming traffic. Output parameter is the *CID* representing the local endpoint. On creation, the group is empty but incoming traffic destined for the PSM value is readable.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Protocol/service multiplexer value

Output Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

7.10 GROUP CLOSE

Service	Input Parameters	Output Parameters
L2CA_GroupClose	CID	Result

Description:

The use of this primitive closes down a Group.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful closure of the channel
0x0001	Invalid CID

7.11 GROUP ADD MEMBER

Service	Input Parameters	Output Parameters
L2CA_GroupAddMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the addition of a member to a group. The input parameter includes the CID representing the group and the BD_ADDR of the group member to be added. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Remote device address

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure to establish connection to remote device
Other	Reserved

7.12 GROUP REMOVE MEMBER

Service	Input Parameters	Output Parameters
L2CA_GroupRemoveMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the removal of a member from a group. The input parameters include the CID representing the group and BD_ADDR of the group member to be removed. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address device to be removed

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure – device not a member of the group
Other	Reserved

7.13 GET GROUP MEMBERSHIP

Service	Input Parameters	Output Parameters
L2CA_GroupMembership	CID	Result, N, BD_ADDR_Lst

Description:

The use of this primitive requests a report of the members of a group. The input parameter CID represents the group being queried. The output parameter Result confirms the success or failure of the operation. If the Result is successful, BD_ADDR_Lst is a list of the Bluetooth addresses of the N members of the group.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure – group does not exist
Other	Reserved

N *Type: uint* *Size: 2 octets*

Value	Description
0x0000-0xFFFF	The number of devices in the group identified by the channel end-point CID. If Result indicates failure, N should be set to 0

BD_ADDR_List *Type: pointer* *Size: N/A*

Value	Description
0xXXXXXXXXXXXX	List of N unique Bluetooth addresses of the devices in the group identified by the channel end-point CID. If Result indicates failure, the all-zero address is the only address that should be returned

7.14 PING

Service	Input Parameters	Output Parameters
L2CA_Ping	BD_ADDR, ECHO_DATA, Length	Result, ECHO_DATA, Size

Description:

This primitive represents the initiation of an L2CA_EchoReq command and the reception of the corresponding L2CA_EchoRsp command.

Input Parameters:

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device.

ECHO_DATA *Type: pointer* *Size: N/A*

Value	Description
N/A	The buffer containing the contents to be transmitted in the data payload of the Echo Request command.

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the data in the buffer.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response received.
0x0001	Timeout occurred.

ECHO_DATA *Type: pointer* *Size: N/A*

Value	Description
N/A	The buffer containing the contents received in the data payload of the Echo Response command.

Size *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the data in the buffer.

7.15 GETINFO

Service	Input Parameters	Output Parameters
L2CA_GetInfo	BD_ADDR, InfoType	Result, InfoData, Size

Description:

This primitive represents the initiation of an L2CA_InfoReq command and the reception of the corresponding L2CA_InfoRsp command.

Input Parameters:

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

InfoType *Type: unit* *Size: 2 octets*

Value	Description
0x0001	Maximum connectionless MTU size

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response received
0x0001	Not supported
0x0002	Informational PDU rejected, not supported by remote device
0x0003	Timeout occurred

InfoData *Type: pointer* *Size: N/A*

Value	Description
N/A	The buffer containing the contents received in the data payload of the Information Response command.

Size *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the data in the InfoData buffer.

7.16 DISABLE CONNECTIONLESS TRAFFIC

Service	Input Parameters	Output Parameters
L2CA_DisableCLT	PSM	Result

Description:

General request to disable the reception of connectionless packets. The input parameter is the *PSM* value indicating service that should be blocked. This command may be used to incrementally disable a set of PSM values. The use of the 'invalid' PSM 0x0000 blocks all connectionless traffic. The output parameter *Result* indicates the success or failure of the command. A limited device might support only general blocking rather than PSM-specific blocks and would fail to block a single non-zero PSM value.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Block all connectionless traffic
0xXXXX	Protocol/Service Multiplexer field to be blocked

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful
0x0001	Failure – not supported

7.17 ENABLE CONNECTIONLESS TRAFFIC

Service	Input Parameters	Output Parameters
L2CA_EnableCLT	PSM	Result

Description:

General request to enable the reception of connectionless packets. The input parameter is the *PSM* value indicating the service that should be unblocked. This command may be used to incrementally enable a set of PSM values. The use of the 'invalid' PSM 0x0000 enables all connectionless traffic. The output parameter *Result* indicates the success or failure of the command. A limited device might support only general enabling rather than PSM-specific filters, and would fail to enable a single non-zero PSM value.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Enable all connectionless traffic
0xXXXX	Protocol/Service Multiplexer field to enable

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful
0x0001	Failure – not supported

8 SUMMARY

The Logical Link Control and Adaptation Protocol (L2CAP) is one of two link level protocols running over the Baseband. L2CAP is responsible for higher level protocol multiplexing, MTU abstraction, group management, and conveying quality of service information to the link level.

Protocol multiplexing is supported by defining channels. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol.

L2CAP abstracts the variable-sized packets used by the Baseband Protocol (page 33). It supports large packet sizes up to 64 kilobytes using a low-overhead segmentation-and-reassembly mechanism.

Group management provides the abstraction of a group of units allowing more efficient mapping between groups and members of the Bluetooth piconet. Group communication is connectionless and unreliable. When composed of only a pair of units, groups provide connectionless channel alternative to L2CAP's connection-oriented channel.

L2CAP conveys QoS information across channels and provides some admission control to prevent additional channels from violating existing QoS contracts.

9 REFERENCES

[1] Internet Engineering Task Force, "A Proposed Flow Specification", RFC 1363, September 1992.

10 LIST OF FIGURES

Figure 1.1: L2CAP within protocol layers 249
Figure 1.2: ACL Payload Header for single-slot packets....250
Figure 1.3: ACL Payload Header for multi-slot packets 250
Figure 1.4: L2CAP in Bluetooth Protocol Architecture 251
Figure 2.1: Channels between devices 254
Figure 2.2: L2CAP Architecture....255
Figure 2.3: L2CAP SAR Variables....255
Figure 2.4: L2CAP segmentation 256
Figure 2.5: Segmentation and Reassembly Services in a unit with an HCI 257
Figure 3.1: L2CAP Layer Interactions 258
Figure 3.2: MSC of Layer Interactions....259
Figure 3.3: State Machine Example 270
Figure 3.4: Message Sequence Chart of Basic Operation 271
Figure 4.1: L2CAP Packet (field sizes in bits) 272
Figure 4.2: Connectionless Packet....273
Figure 5.1: Signalling Command Packet Format....275
Figure 5.2: Command format 275
Figure 5.3: Command Reject Packet 277
Figure 5.4: Connection Request Packet....278
Figure 5.5: Connection Response Packet....279
Figure 5.6: Configuration Request Packet 281
Figure 5.7: Configuration Request Flags field format....281
Figure 5.8: Configuration Response Packet....283
Figure 5.9: Configuration Response Flags field format....283
Figure 5.10: Disconnection Request Packet 285
Figure 5.11: Disconnection Response Packet 286
Figure 5.12: Echo Request Packet 286
Figure 5.13: Echo Response Packet....287
Figure 5.14: Information Request Packet....287
Figure 5.15: Information Response Packet....288
Figure 6.1: Configuration option format....289
Figure 6.2: MTU Option Format 290
Figure 6.3: Flush Timeout 290
Figure 6.4: Quality of Service Flow Specification 291
Figure 6.5: Configuration State Machine....294

Figure I Basic MTU exchange 318
Figure II Dealing with Unknown Options 319
Figure III Unsuccessful Configuration Request 320

11 LIST OF TABLES

Table 1.1: Logical channel L_CH field contents 250
Table 2.1: CID Definitions 253
Table 2.2: Types of Channel Identifiers 254
Table 3.1: L2CAP Channel State Machine 267
Table 5.1: Signalling Command Codes 276
Table 5.2: Reason Code Descriptions 277
Table 5.3: Reason Data values 278
Table 5.4: Defined PSM Values 278
Table 5.5: Result values 280
Table 5.6: Status values 280
Table 5.7: Configuration Response Result codes 284
Table 5.8: InfoType definitions 287
Table 5.9: Information Response Result values 288
Table 5.10: Information Response Data fields 288
Table 6.1: Service type definitions 292
Table 6.2: Parameters allowed in Request 293
Table 6.3: Parameters allowed in Response 294

Table I Result of Second Link Timeout Request 322
Table II Result of Second Flush Timeout Request 322

TERMS AND ABBREVIATIONS

Baseband	Baseband Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IrDA	Infra-red Data Association
L_CH	Logical Channel
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MTU	Maximum Transmission Unit
PPP	Point-to-Point Protocol
Reliable	Characteristic of an L2CAP channel that has an infinite flush timeout
RFC	Request For Comments
SAR	Segmentation and Reassembly

APPENDIX A: CONFIGURATION MSCs

The examples in this appendix describe a sample of the multiple possible configuration scenarios that might occur. Currently, these are provided as suggestions and may change in the next update of the Specification.

Figure I illustrates the basic configuration process. In this example, the devices exchange MTU information. All other values are assumed to be default.

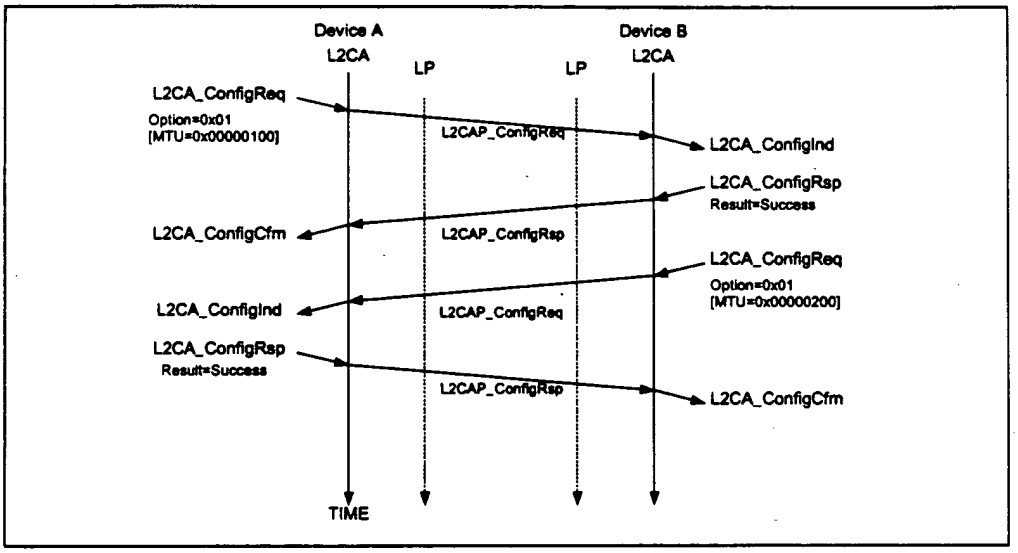


Figure I: Basic MTU exchange

Figure II on page 319 illustrates how two devices interoperate even though one device supports more options than the other does. Device A is an upgraded version. It uses a hypothetically defined option type 0x20 for link-level security. Device B rejects the command using the Configuration Response packet with result 'unknown parameter' informing Device A that option 0x20 is not understood. Device A then resends the request omitting option 0x20. Device B notices that it does not need to such a large MTU and accepts the request but includes in the response the MTU option informing Device A that Device B will not send an L2CAP packet with a payload larger than 0x80 octets over this channel. On receipt of the response, Device A could reduce the buffer allocated to hold incoming traffic.

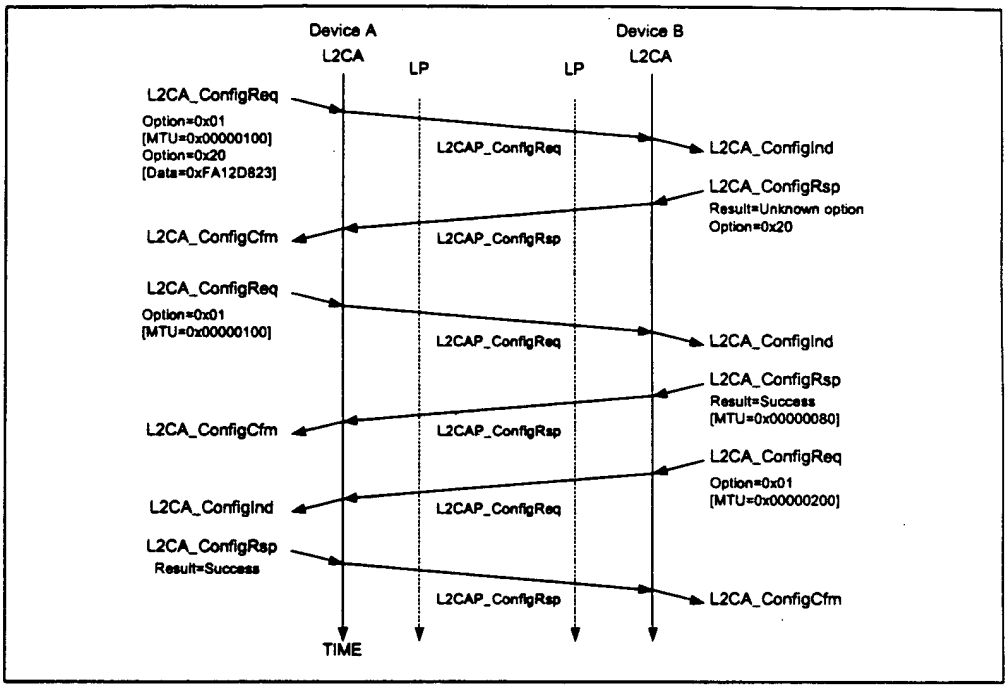


Figure II: Dealing with Unknown Options

Figure III on page 320 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is that the configuration request is placed in an L2CAP packet that cannot be accepted by the remote device, due to its size. The remote device informs the sender of this problem using the Command Reject message. Device A then resends the configuration options using two smaller L2CAP_ConfigReq messages.

The second problem is an attempt to configure a channel with an invalid CID. For example device B may not have an open connection on that CID (0x01234567 in this example case).

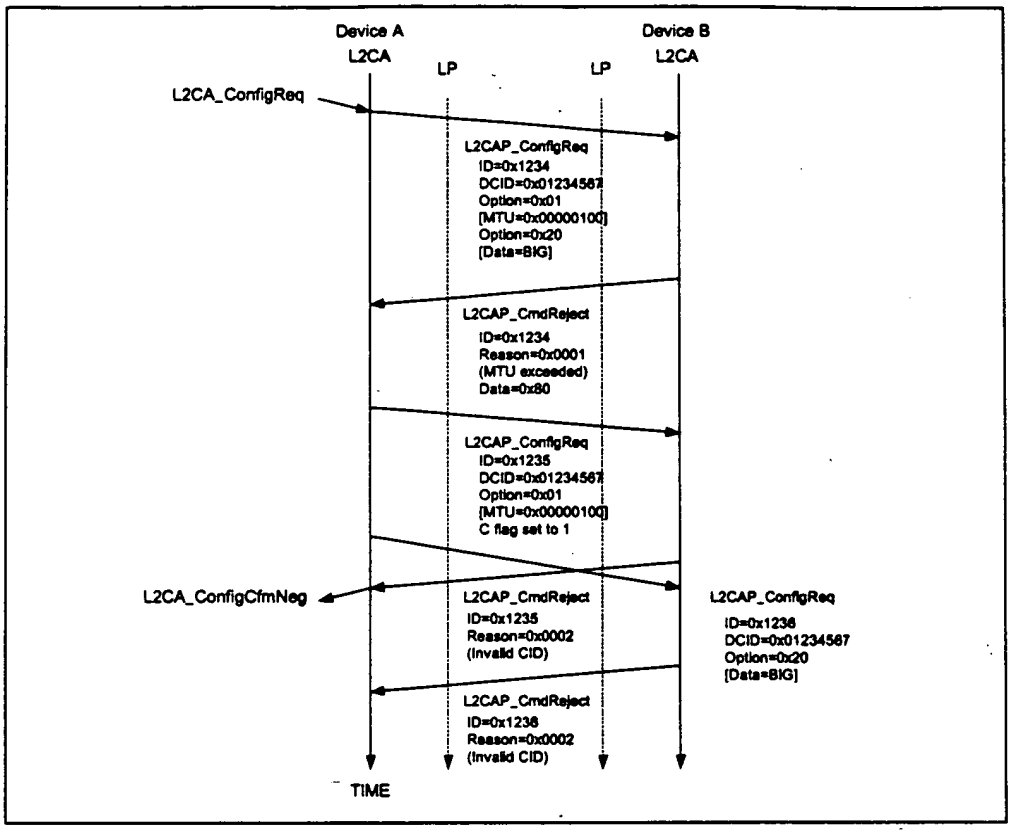


Figure III: Unsuccessful Configuration Request

APPENDIX B: IMPLEMENTATION GUIDELINES

This section contains some guidelines for implementations. These guidelines are not part of the compliance tests. At the moment they are simply suggestions on how to solve some difficult problems.

RTX TIMER

Implementations should not start this timer on an L2CAP Connection Request packet unless the physical link has been established. Otherwise the Baseband paging mechanism might increase the cost of the request beyond that of the minimal timeout value. If an implementation performs some form of security check it is recommended that the connection pending response be sent back prior to any consultation with a security manager that might perform Baseband authentication commands. If any security check requires user interaction, the link might timeout waiting for the user to enter a PIN.

QOS MAPPING TO LM AND L2CAP IMPLEMENTATIONS

Token Rate

The Link Manager (LM) should ensure data is removed from the transmission buffer at this rate. The LM should ensure the polling interval is fast enough to support this data rate. The polling interval should be adjusted if the packet type changes. If the buffer overflows, and the service type is Guaranteed, a QoS violation should be reported. If the service type is Best Effort, and a Token Rate was non-zero, a QoS violation should also be reported.

Given a Token Rate of 0xFFFFFFFF, and Service Type of Guaranteed, the LM should refuse any additional connections from remote devices and disable all periodic scans.

Token Bucket Size

L2CAP implementations should ensure that a buffer meeting the size request is allocated for the channel. If no buffer is available, and the service type is Guaranteed, the request should be rejected. If no appropriately sized buffer is available, and the service type is Best Effort, the largest available buffer should be allocated.

Peak Bandwidth

If the token bucket buffer overflows, a QoS violation should be raised.

Latency

The LM should ensure the polling interval is at least this value. If the polling interval necessary to support the token rate is less than this value, the smaller interval should be used. If this interval cannot be supported, a QoS violation should be raised.

Delay Variation

The LM may ignore this value because there is no clear mapping between L2CAP packet delays and the necessary polling interval without requiring the LM to comprehend the length field in L2CAP packets.

COLLISION TABLES

Current Value	Requested Value	Result
X	X	X
X	Y	If (X<Y) then X else Y

Table I: Result of Second Link Timeout Request

Current Value	Requested Value	Result
N	0	N
N	N	N
N	M != N	Reject

Table II: Result of Second Flush Timeout Request

APPENDIX B

Windows Wireless Architecture

Tim Moore
Lead Program Manager
Windows Networking
Microsoft Corporation

0956565 .0424000

Agenda

- Wireless trends
 - WAN, LAN, PAN
- Scenarios
 - Adhoc, home, small business
 - Enterprise, ISP
- Wireless architecture
- Summary
- Call to action
- More information

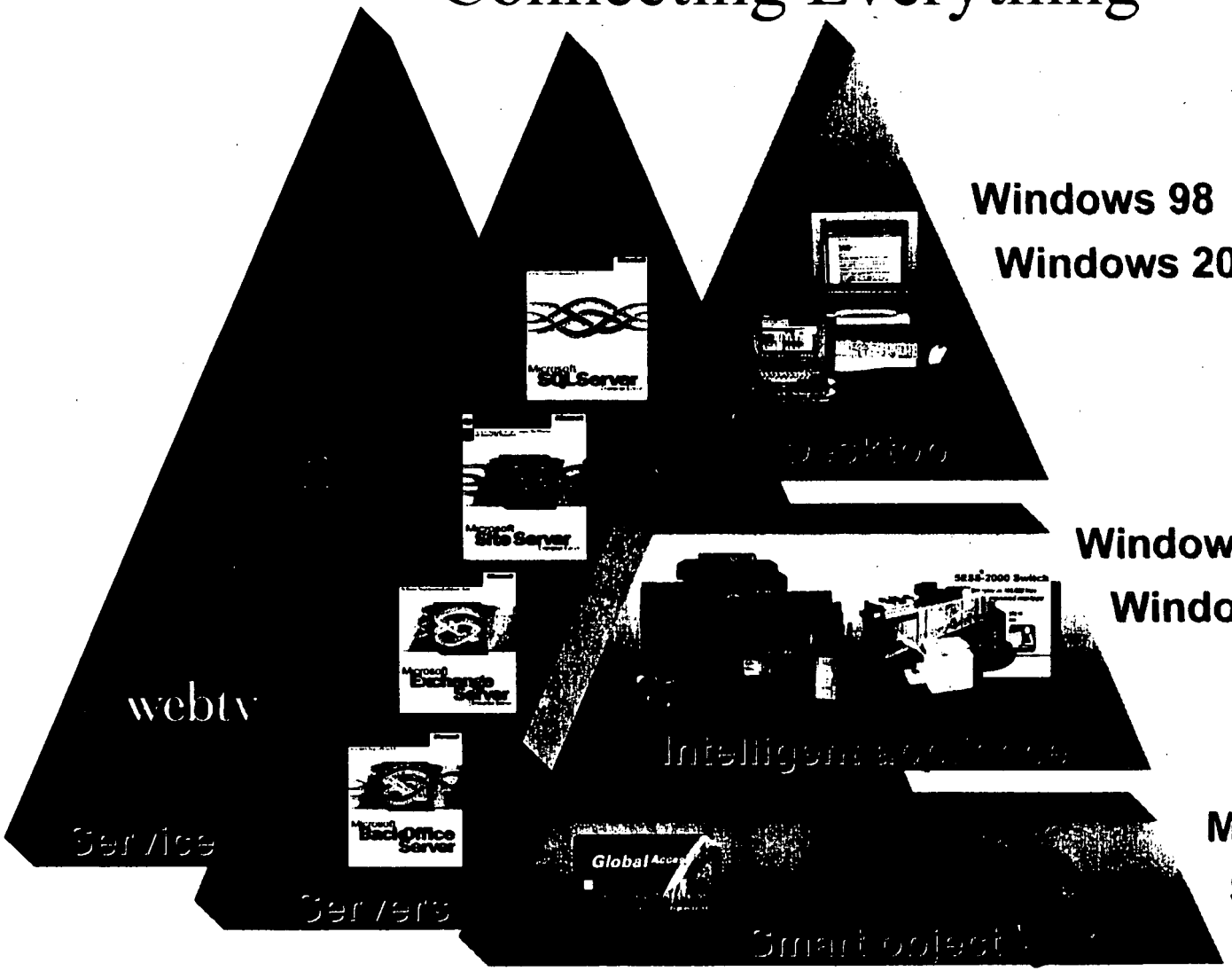
0056565 .042400

Wireless Trends

- IP networks
- Always connected
- Increased bandwidth
- Convenience
- Moving from vertical market to horizontal markets
- Moving from proprietary to standards based
- Proliferation of smart devices
- New scenarios enabled
- Outsourcing
- Adhoc networks

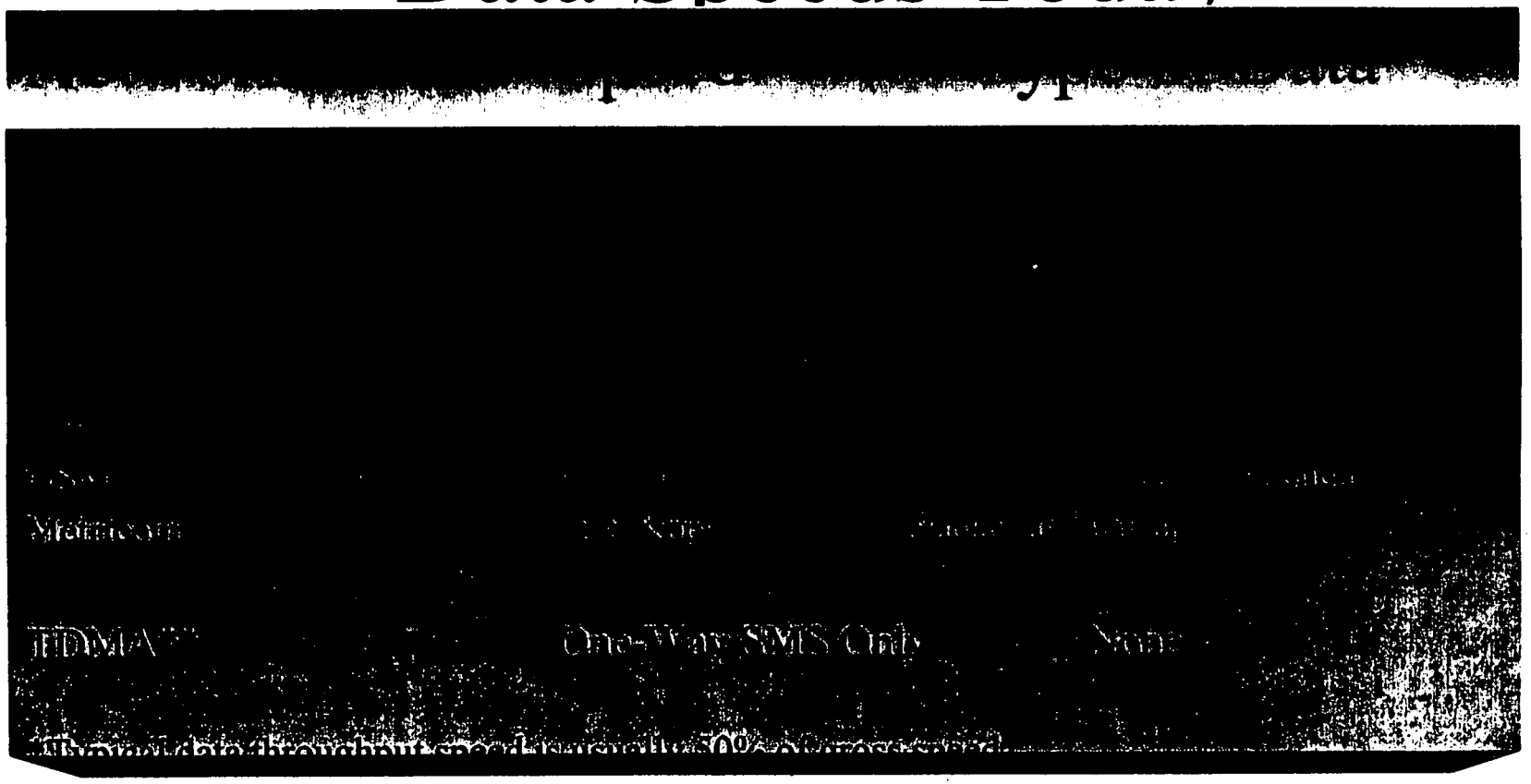
Information Anytime, Anywhere

Connecting Everything



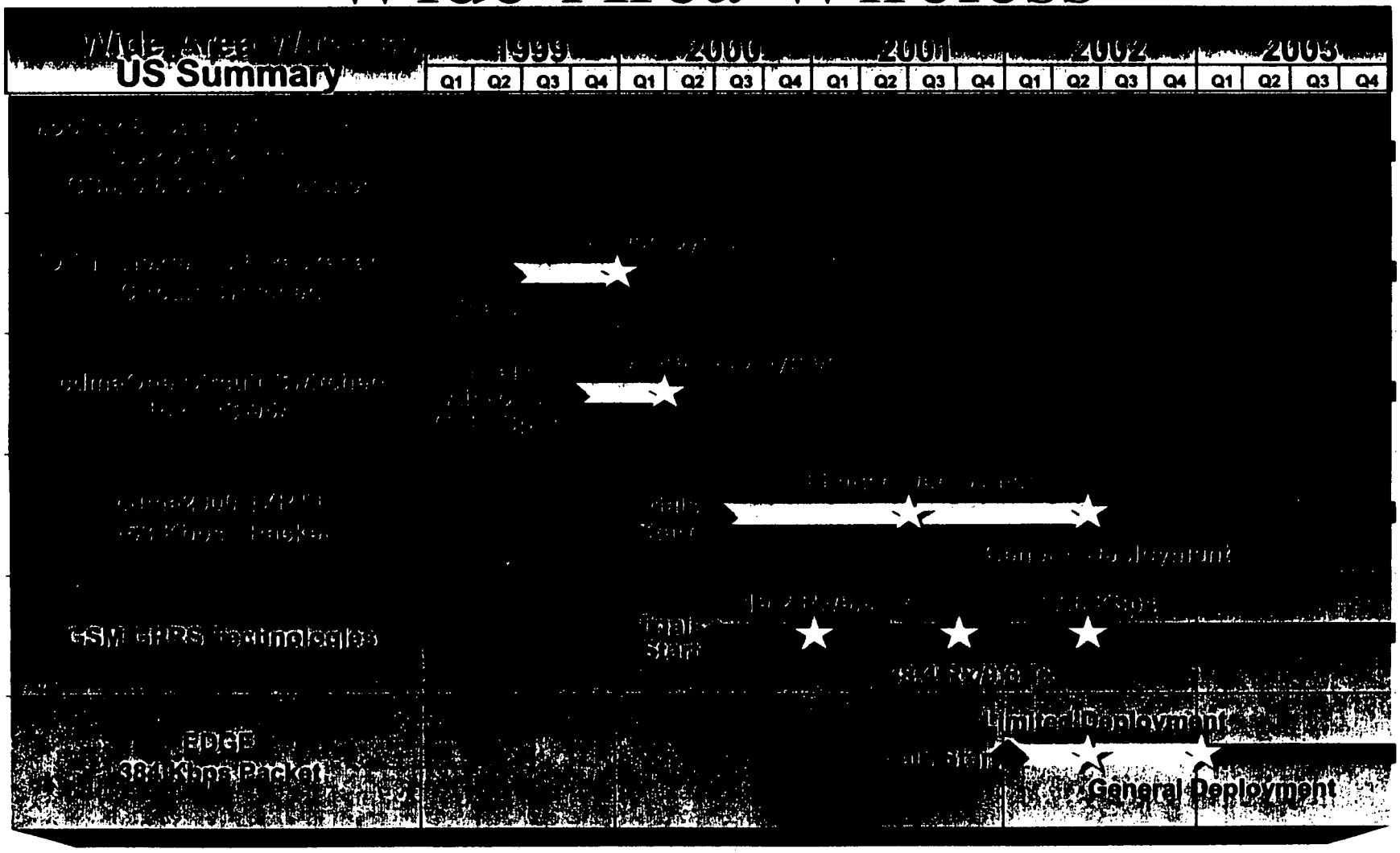
0056656S .012400

Data Speeds Today



**TDMA systems do not support data in the U.S. at this time

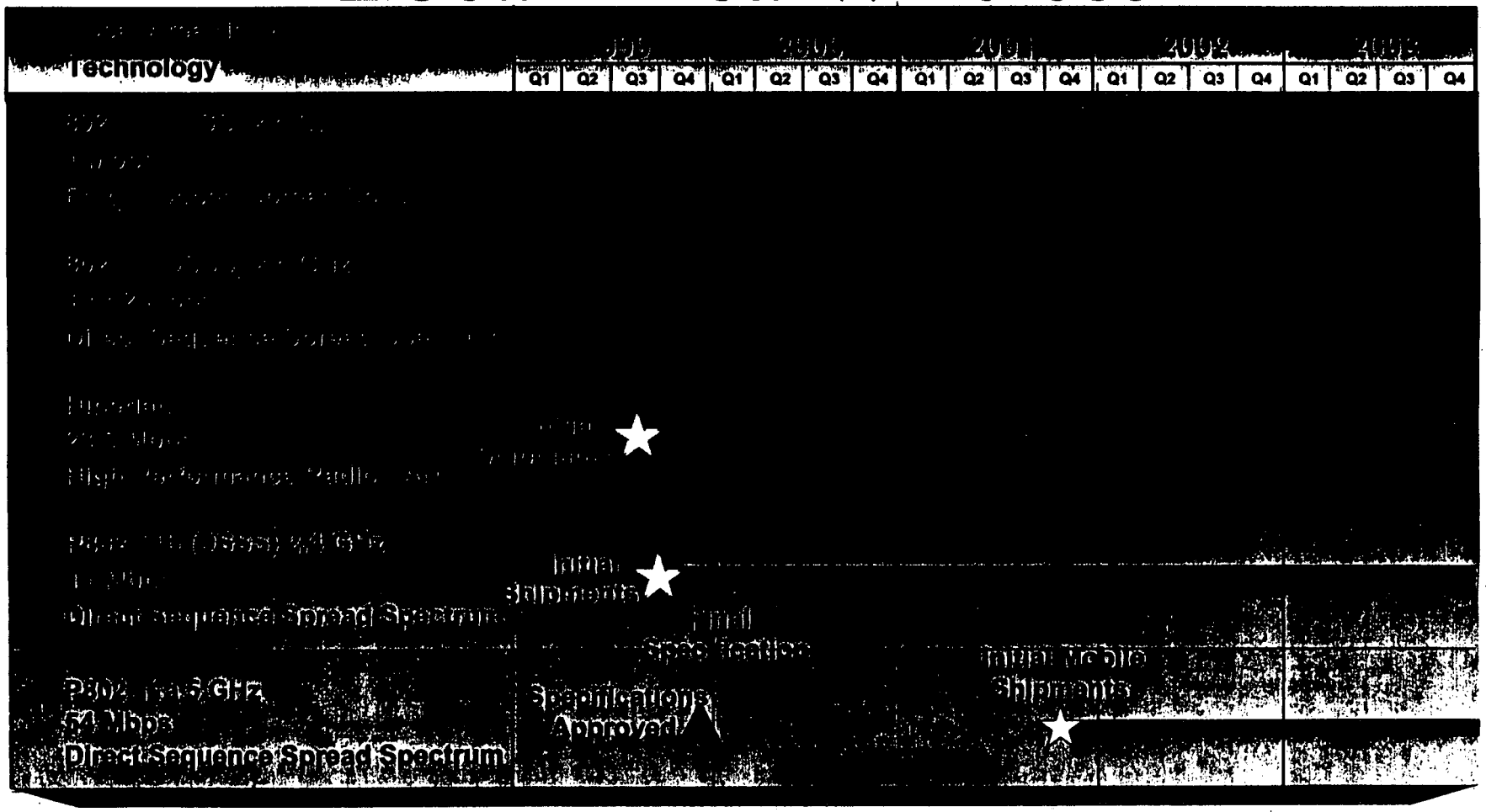
Wide-Area Wireless



Materials from Andrew Seybold-Microsoft Exchange Conference 1999

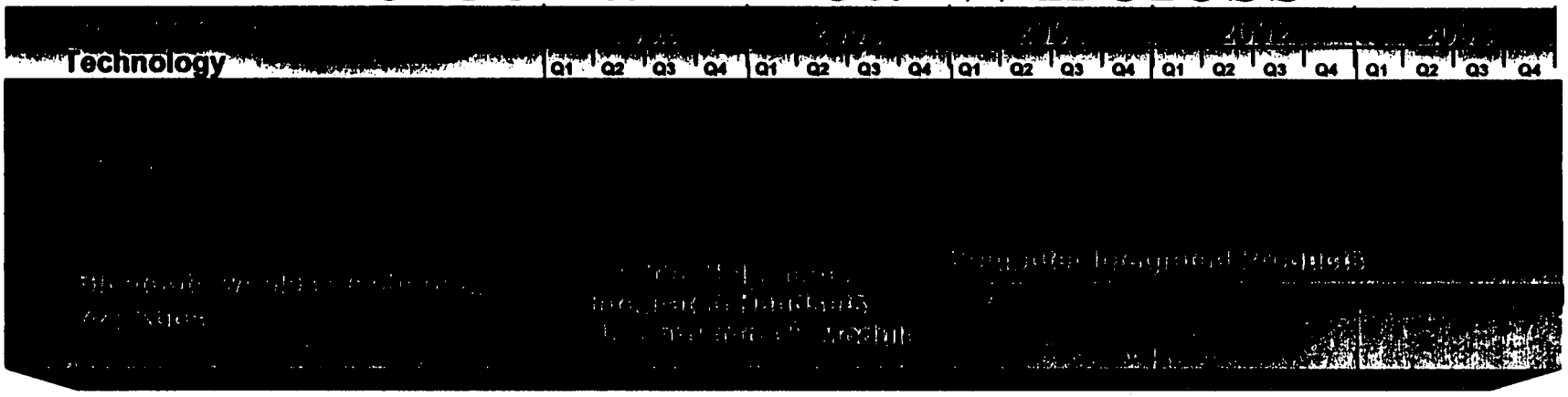
095S6S6S6S • 01210D0

Local-Area Wireless



Materials from Andrew Seybold-Microsoft Exchange Conference 1999

Personal Area Wireless



Personal Area Wireless

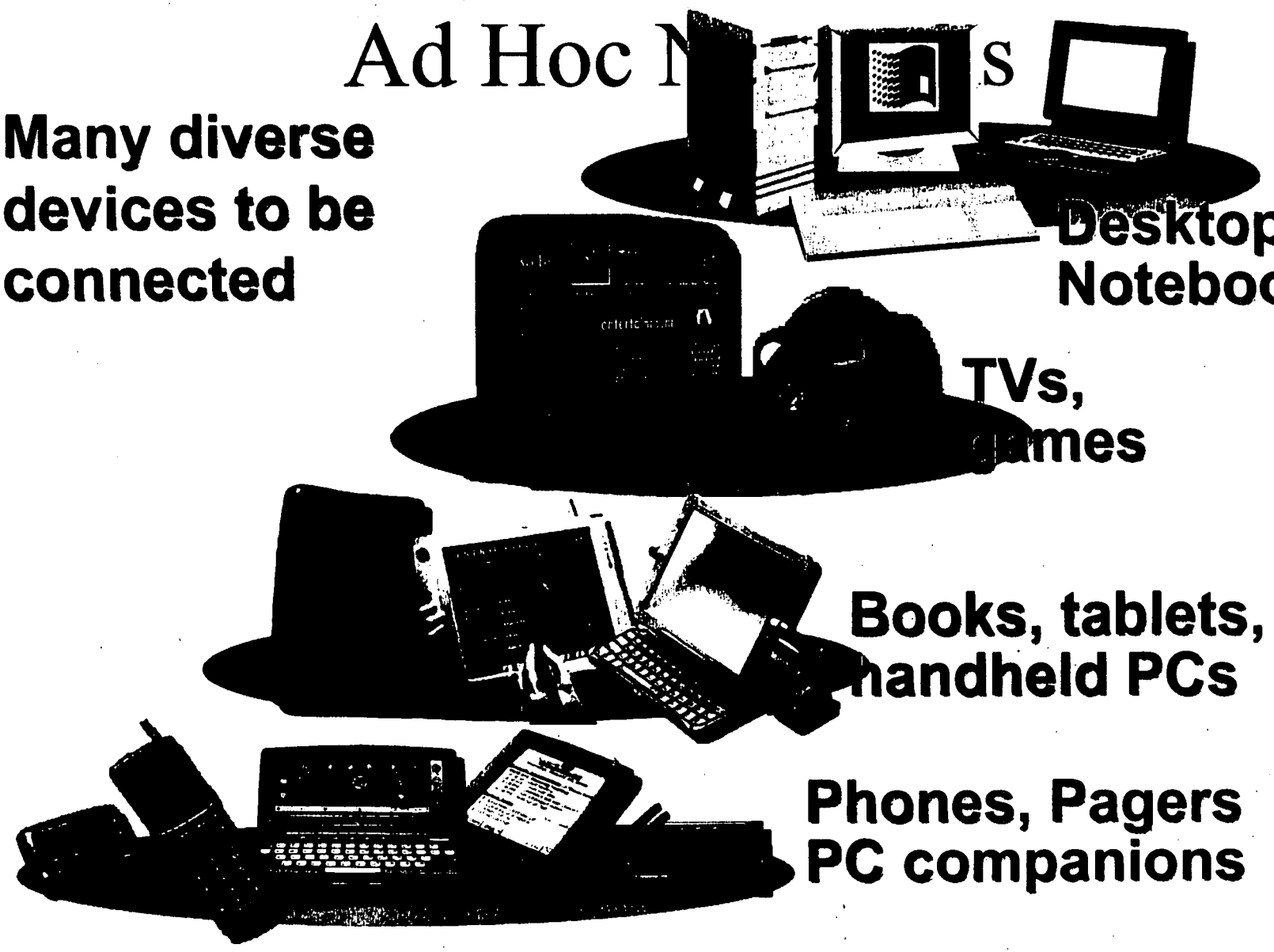
- IrDA
 - Around since 1994
 - Available on every PC and lots of devices
 - >20 million existing IrDA devices
 - Camera, PDAs, cellphones, printers, keyboards
- Exploding market fueled by Bluetooth momentum
 - Bluetooth wireless technology is a defacto standard
 - Proliferation of smart devices, convenience of cable replacement, and new usage scenarios

Scenarios

- Adhoc
- Home
- Small business
- Enterprise
- ISP

Ad Hoc N[illegible]s

- **Many diverse devices to be connected**



Desktops, Notebooks

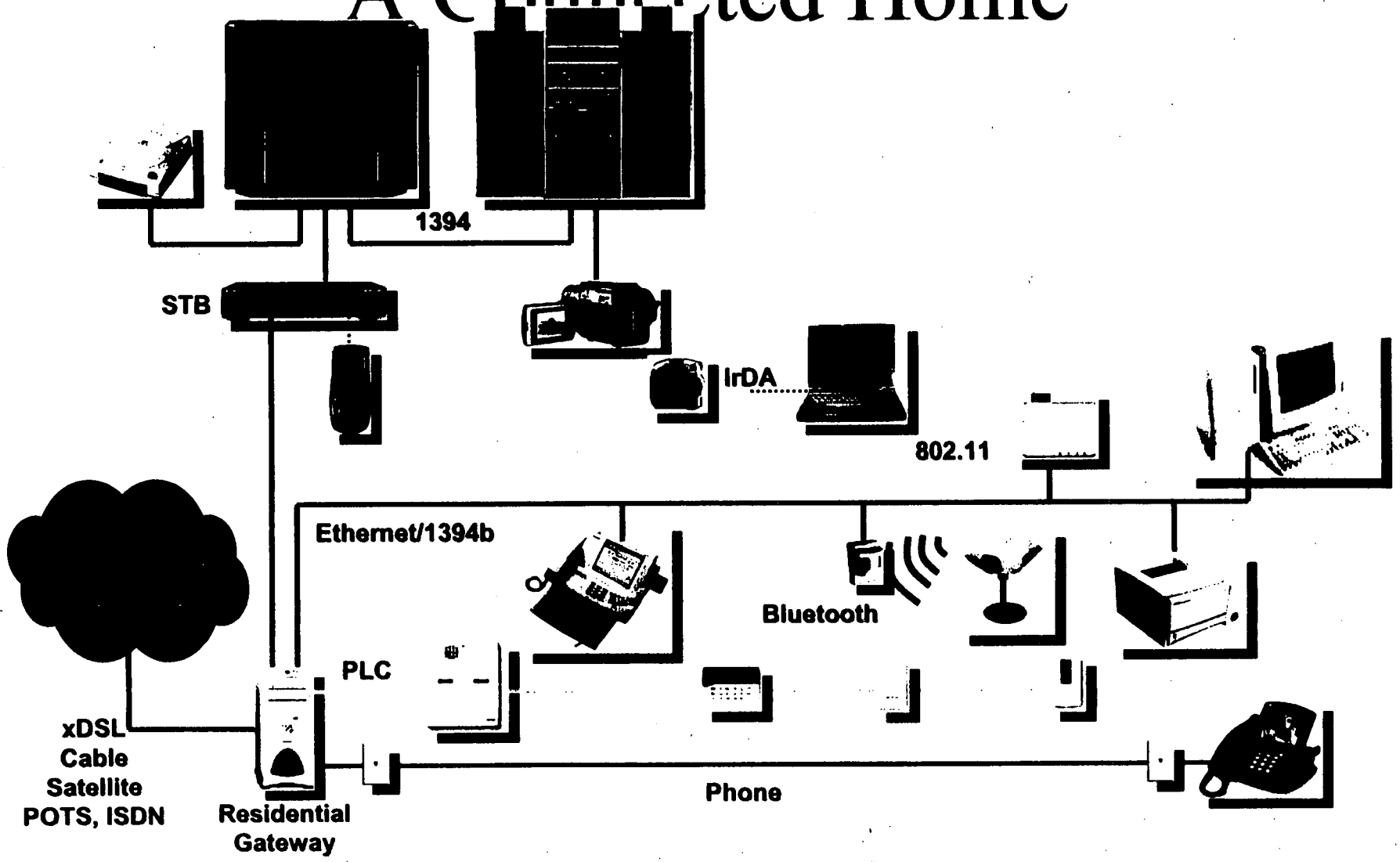
TVs, [illegible]mes

Books, tablets, handheld PCs

Phones, Pagers PC companions

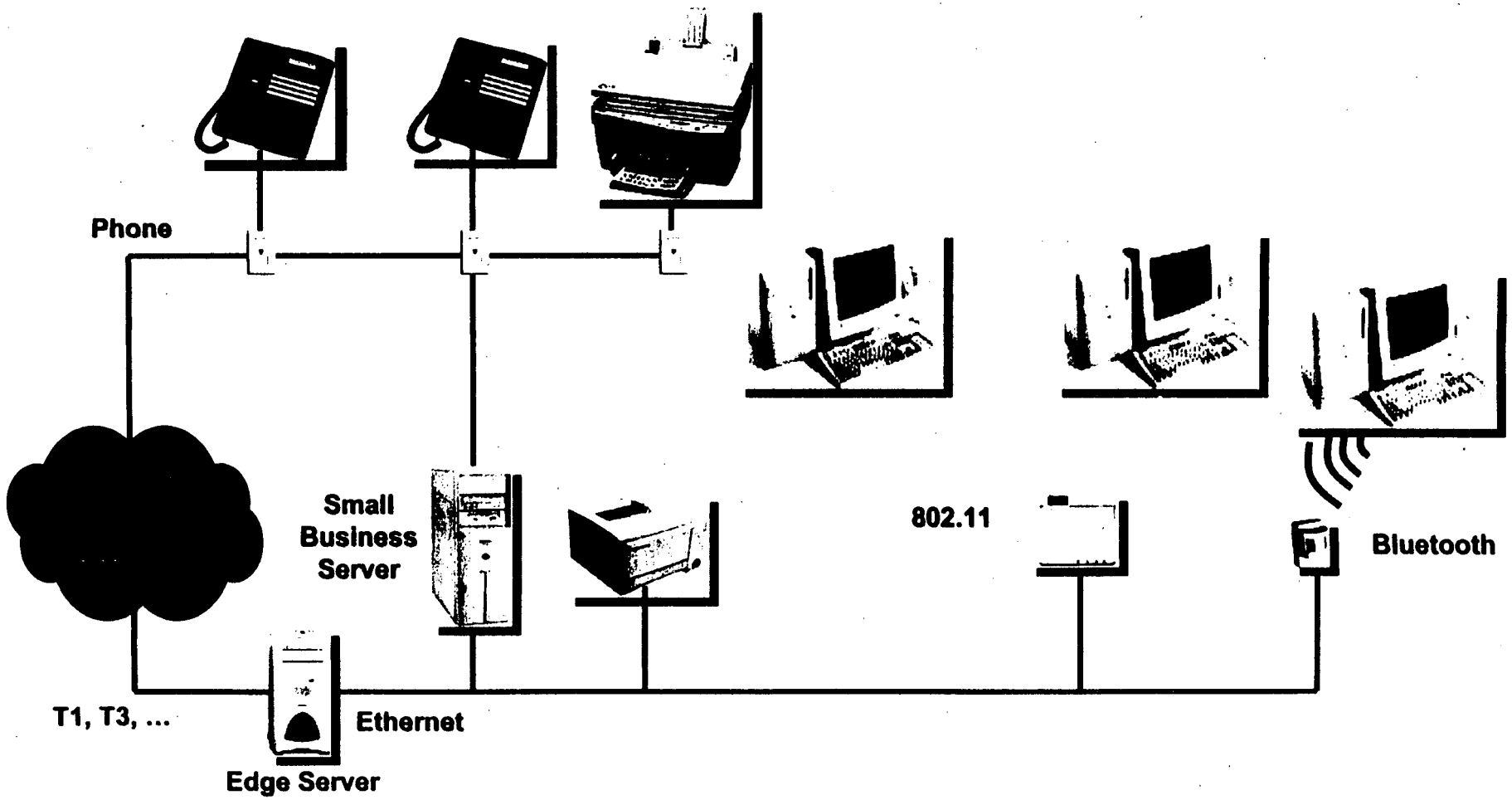
0956565 .0121109

A Connected Home



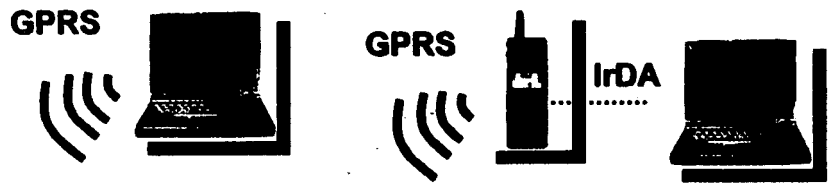
D9556565.012400

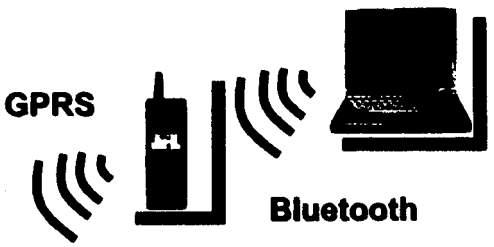
A Connected Small Office



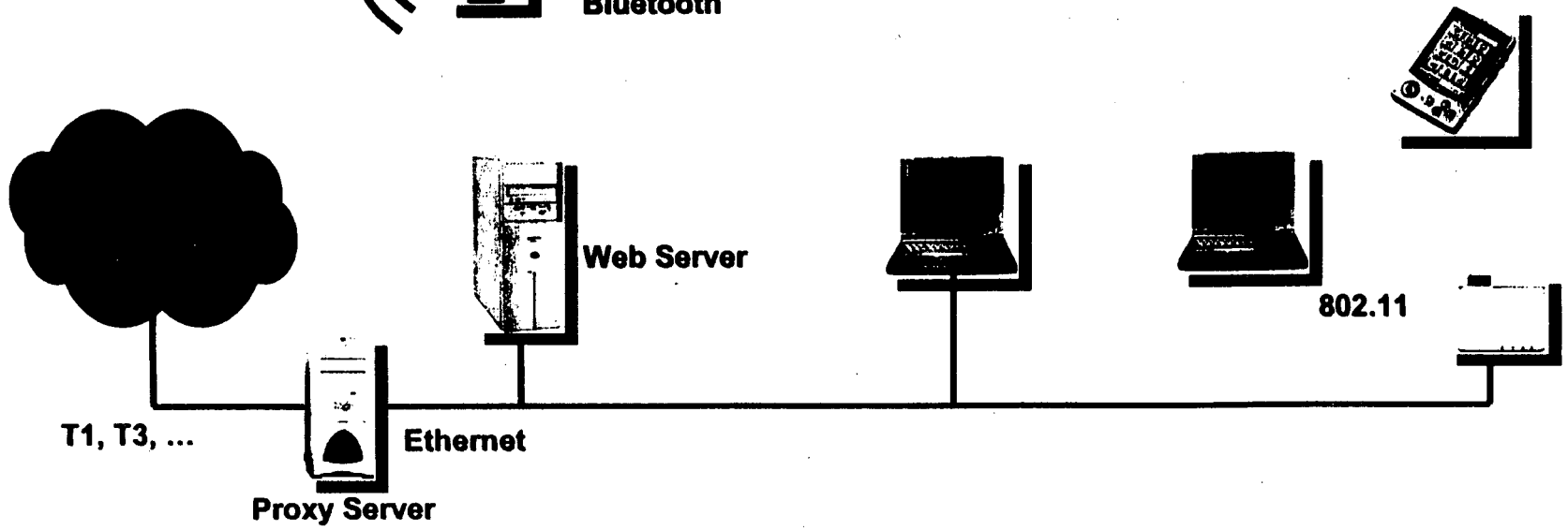
0055655 .01310

Enterprise





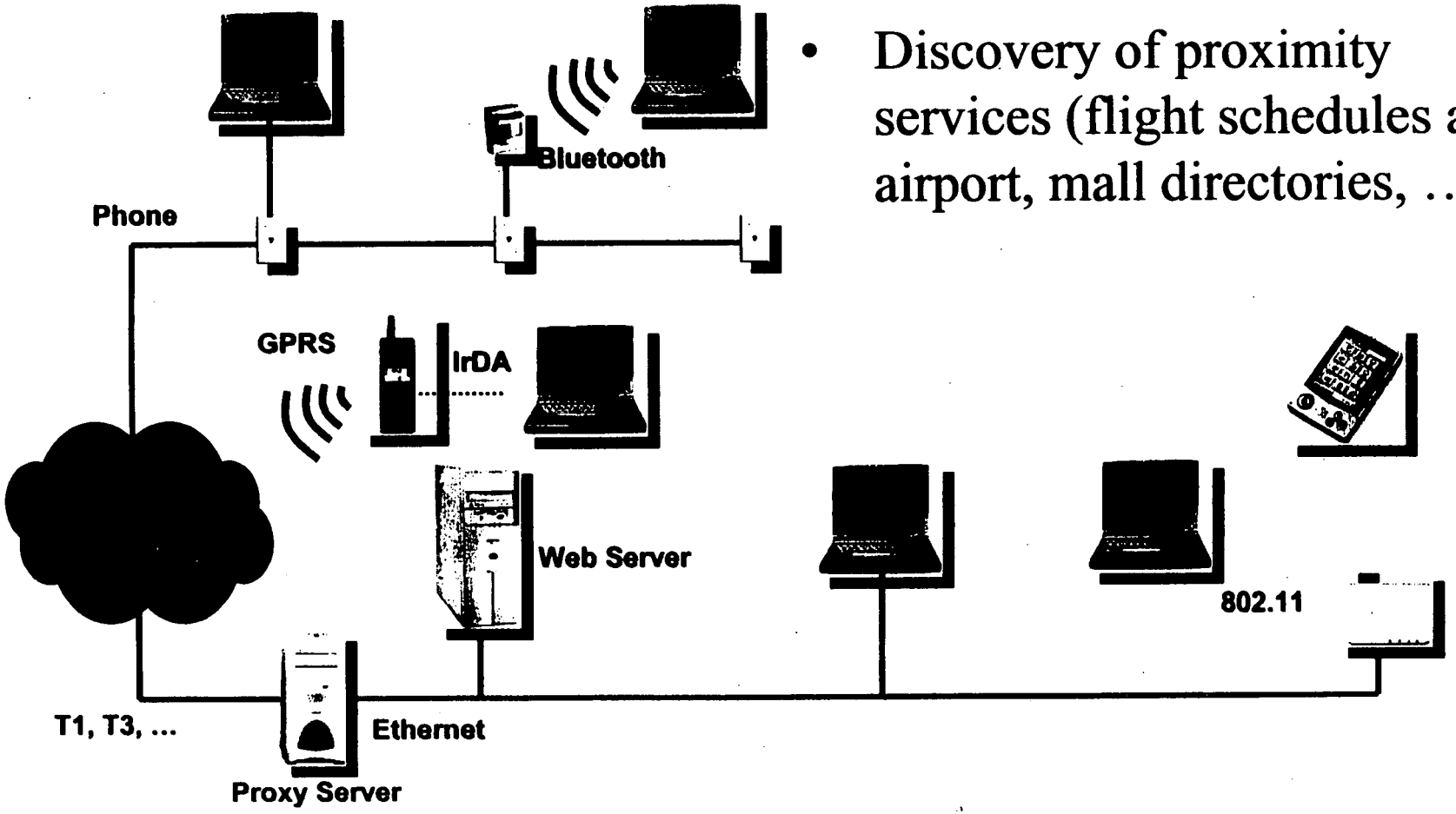
- Information at your fingertips
 - At meetings, in the office, on the road
 - Reliable, secure, multimedia LAN



Enterprise

- End-user can access the enterprise wireless network transparently over a secure connection
 - The network administrator has control over which users have access to the enterprise wireless LAN
- Enterprise can offer its employees access via ISPs which outsource their authentication to the enterprise
 - End-user has IP connectivity as soon as a CDPD or a GPRS modem is plugged in
 - Make cellphones an always connected Internet access point using GPRS
- End-User can use Netmeeting with wireless LAN, when out of range of LAN can continue to conference via IP connected cellphone

An ISP Connected Public Space



- Discovery of proximity services (flight schedules at airport, mall directories, …)

ISP

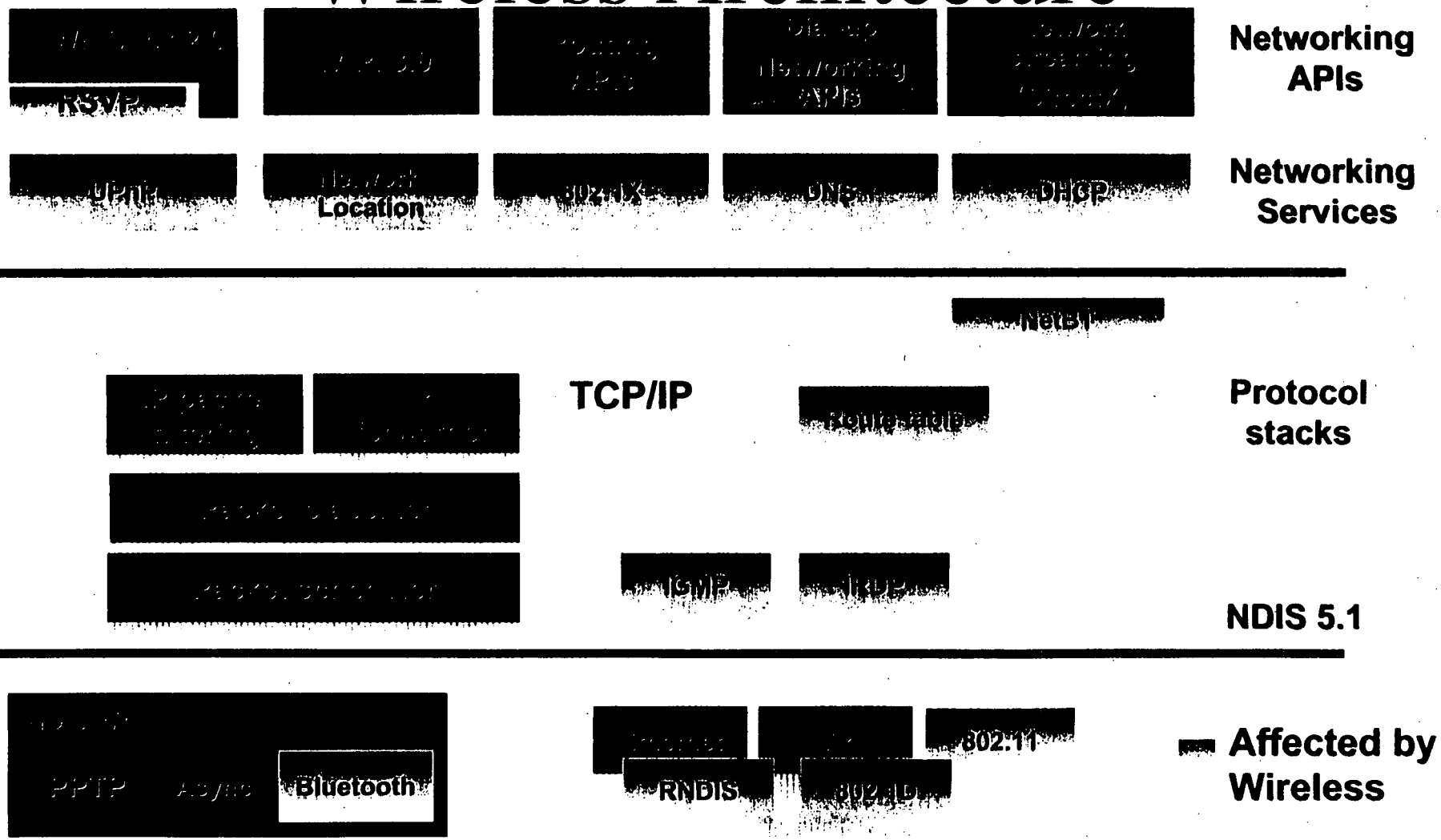
- Need mixed technologies
 - Higher speed in hot spots, e.g., 802.11
- Need authentication so ISPs can charge
- Allow ISPs to integrate into existing Radius systems
 - Allows ISP roaming agreements
 - Same as outsource dial
- Need to be able to provision unauthenticated users

Wireless Architecture

- "Just works"
- Always connected
- Unified transport: IP
- Mobility
- Unified security model
- Adhoc
- QoS
- Performance

0955655 .0124100

Wireless Architecture



0955565 .0429.00

Just Works

- No configuration
 - Especially when roaming
- CDPD
 - Configure Network Equipment Identifier
- 802.11
 - Configure network name and security keys
 - Per location
- Bluetooth wireless technology
 - Configure PIN numbers
 - Per device

802.11 Configuration

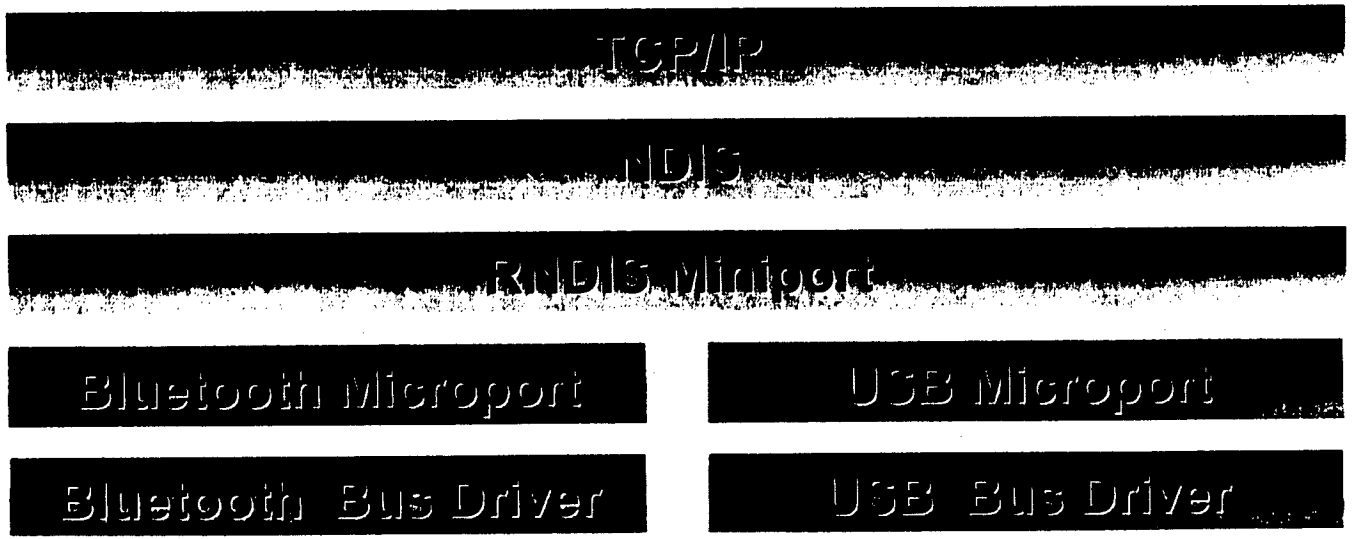
- Current 802.11 networks need to be configured with name of the network
 - Roaming between multiple networks difficult especially when security is implemented
- Automatically find a wireless network
 - If Access point is beaconing network name, attempt to use that network
 - If no infrastructure available then switch to adhoc mode

Always Connected

- Permanent IP connectivity should not use dial-up model
 - A CDPD card should appears as a LAN card
 - A GPRS, EDGE or 3G card or cellphone should appear as a LAN card
 - GPRS Terminal Type Recommendations
 - Cellphone needs to be Type A (voice and packet)
 - PC-Card can be Type C (packet only)
- Implement an NDIS driver or use Remote NDIS
 - Remote NDIS over Bluetooth connections

Remote NDIS

- Remote NDIS enables a bus-agnostic connection to devices that provide network access
- Remote NDIS is both a driver architecture and a command language



Unified Transport: IP

- All other media except Bluetooth wireless technology support always connected IP
- Ethernet over point-to-point Bluetooth connections
 - L2 bridge gives an adhoc L2 network
- Adhoc applications use UPnP over IP
- Expect large numbers of wireless connected devices
 - Move to IPv6 for addresses

Mobility

- Applications should not rely on having a network available all the time
 - Network connection can disappear at anytime
 - Applications should reconnect automatically if the network appears
- Clients hold state about the network
 - IP address
 - Routes
- Networks hold state about the client
 - Multicast distribution
 - Quality of service
 - Secure access
 - Machine name to IP address mapping
- How to detect when this state is out of date
- Applications also hold state about the network
 - TCP connections
 - E.g. Proxies, firewalls, etc.

Mobility

- Detect roaming
 - Mediasense detects working/non-working interfaces
 - Mediasense detects interfaces changing their network connection
- IP address
 - Mediasense triggers a DHCP renew; If renew fails, DHCP gets a new IP address
 - DHCP updates DNS when an address changes
 - TCP/IP removes IP addresses if NIC not connected
 - Mobile IP allows IP address to stay the same when roaming

Mobile IP

- Mobile IP keeps the application IP address the same
 - IPv4 has two options
 - Change the network interface address to a local IP address
 - Use an ARP proxy to keep the same IP address
 - IPv6 only has first option
- Mobile IP Issues
 - How to route efficiently
 - IPv6 fixes this issue
 - Firewall traversal
 - Time to get a local address
 - Doesn't allow Voice over IP roaming
- Doesn't address any of the other issues with multicast, QoS, security, applications
- GPRS and 3G have network layer mobility
- No plans to support Mobile IP until IPv6

Mobility

- Multicast
 - Mediasense triggers IGMP refresh on roaming
- QoS
 - Mediasense triggers RSVP refresh on roaming
- Routes
 - Mediasense triggers router detect (IRDP) on roaming
 - Default interface metrics should depend interface speed
 - Routes to no longer existing interface addresses are removed
- Security
 - Mediasense triggers network authentication refresh
- Applications
 - Need to retry connections on connection failure and mediasense
 - Configurations based on network location

0056565 .0424000

Network Location API

- Network location is a hint to the application of the network the machine is connected to
- Accessible via Winsock API
 - Query for the connected networks
 - WSALookupServiceBegin
 - WSALookupServiceNext
 - WSALookupServiceEnd
 - Request for notification when the connected networks changes
 - WSANSIoctl (,SIO_NSP_NOTIFY_CHANGE,…)
- Applications that need configuration per network should use this API
 - E.g., application proxies

Security

- Secure access to resources in the network
 - This is Windows login
- Secure transfer of data over the network
 - This is IPSec
 - Integrated into Windows credentials using PKI and Kerberos
- Secure access to the network
 - This is available for RAS and VPNs
 - Integrated into Windows credentials using PKI (EAP) and Radius
 - Supports roaming of identities
- No secure access to LAN networks
 - Very important for Wireless

0956565 .0434400

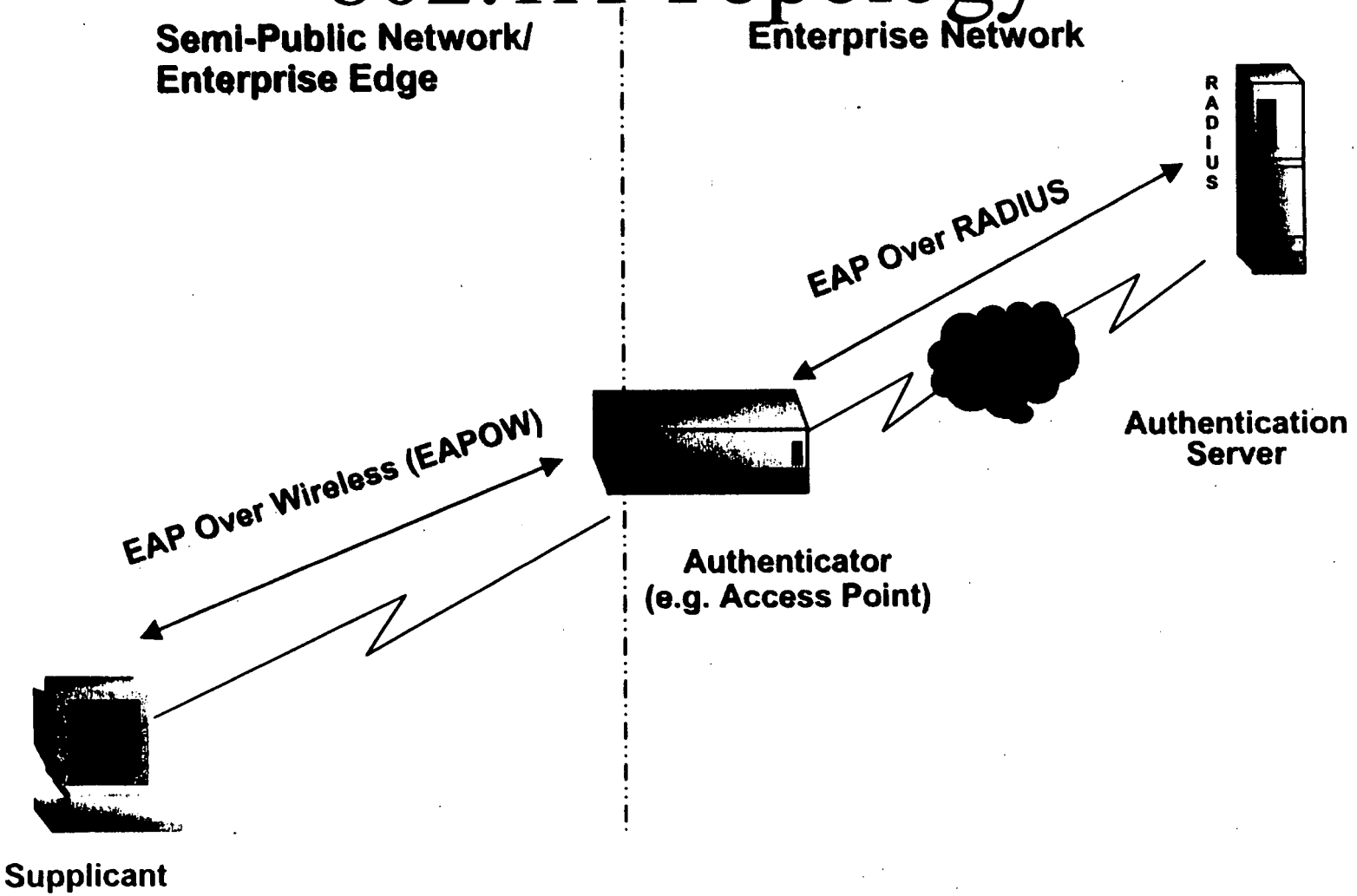
Wireless Security Issues

- User loses wireless NIC, doesn't report it
 - Without user authentication, Intranet now accessible by attackers
 - Without centralized accounting and auditing, no means to detect unusual activity
 - Users who don't log on for periods of time
 - Users who transfer too much data, stay on too long
 - Multiple simultaneous logins
 - Logins from the "wrong" machine account
 - With global keys, large scale re-keying required

Wireless Deployment Issues

- User administration
 - Integration with existing user administration tools required (RADIUS, LDAP-based directories)
 - Create a Windows group for wireless
 - Any user or machine who is a member of the group has wireless access
 - Identification via User-Name easier to administer than MAC address identification
 - Usage accounting and auditing desirable
- Key management
 - Static keys difficult to manage on clients, access points
 - Proprietary key management solutions require separate user databases

802.1X Topology



0955565 .0121403

IEEE 802.1X

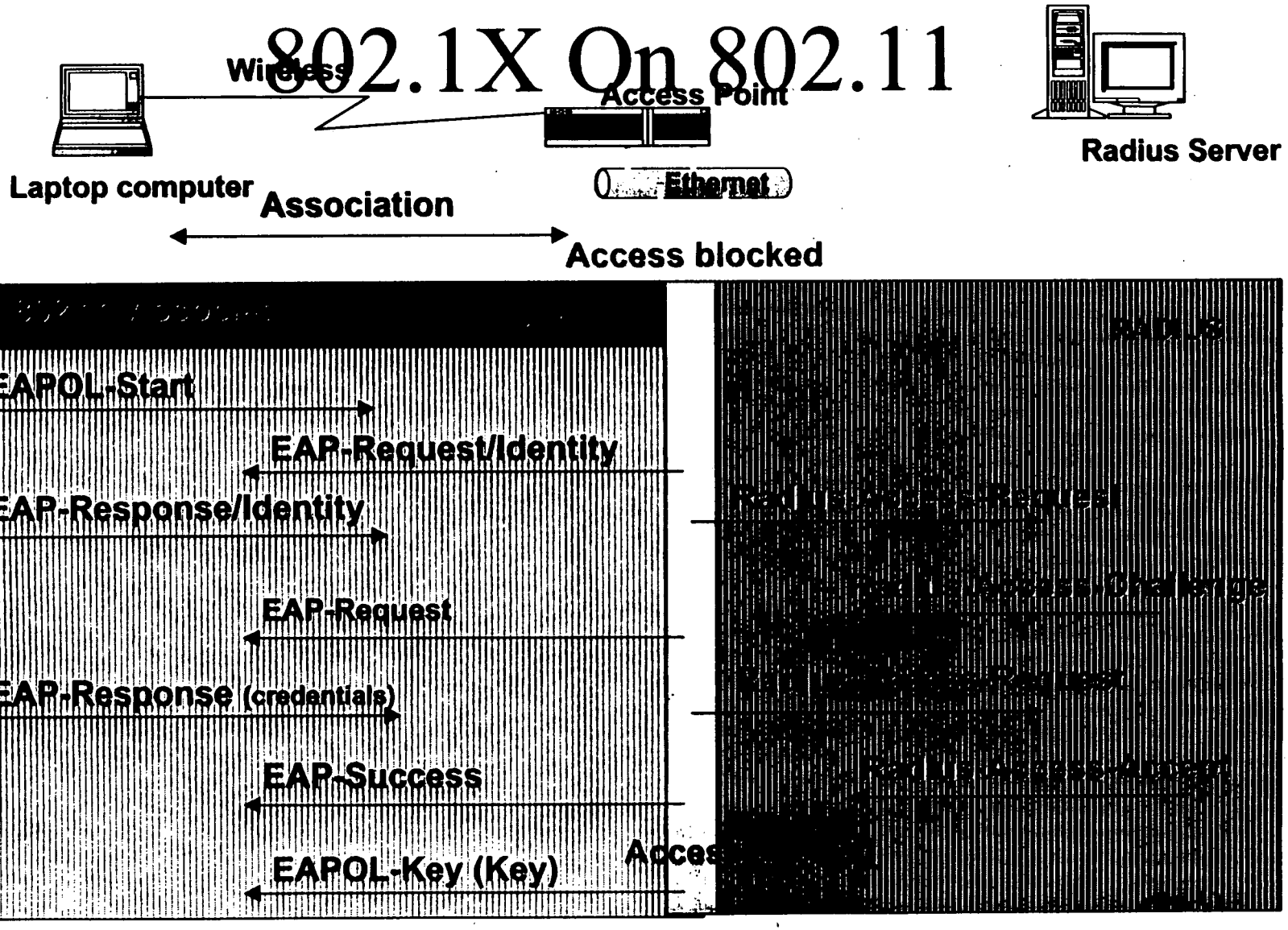
- Enables interoperable user identification, centralized authentication, key management
 - Leverages existing standards: EAP, RADIUS
 - Compatible with existing roaming technologies, enabling use in hotels and public places
- User-based identification
 - Identification based on Network Access Identifier (RFC 2486) enables support for roaming access in public spaces (RFC 2607)
- Dynamic key management
- Centralized user administration
 - Support for RADIUS (RFC 2138, 2139) enables centralized authentication, authorization and accounting
 - RADIUS/EAP (draft-ietf-radius-ext-07.txt) enables encapsulation of EAP packets within RADIUS
- Supported on Ethernet, Token Ring and 802.11

Extensible Authentication Protocol

- Used by PPP for RAS and VPN
- Allows support for a number of authentication mechanisms
 - EAP designed to allow additional authentication methods to be deployed with no changes to the access point or client NIC
 - RFC 2284 includes support for password authentication (EAP-MD5), One-Time Passwords (OTP)
 - Windows 2000 supports smartcard authentication (RFC 2716) and Security Dynamics
- Radius server used for authentication and authorization
 - Integrated into Active Directory™ users and groups
 - Supports cross authentication for roaming

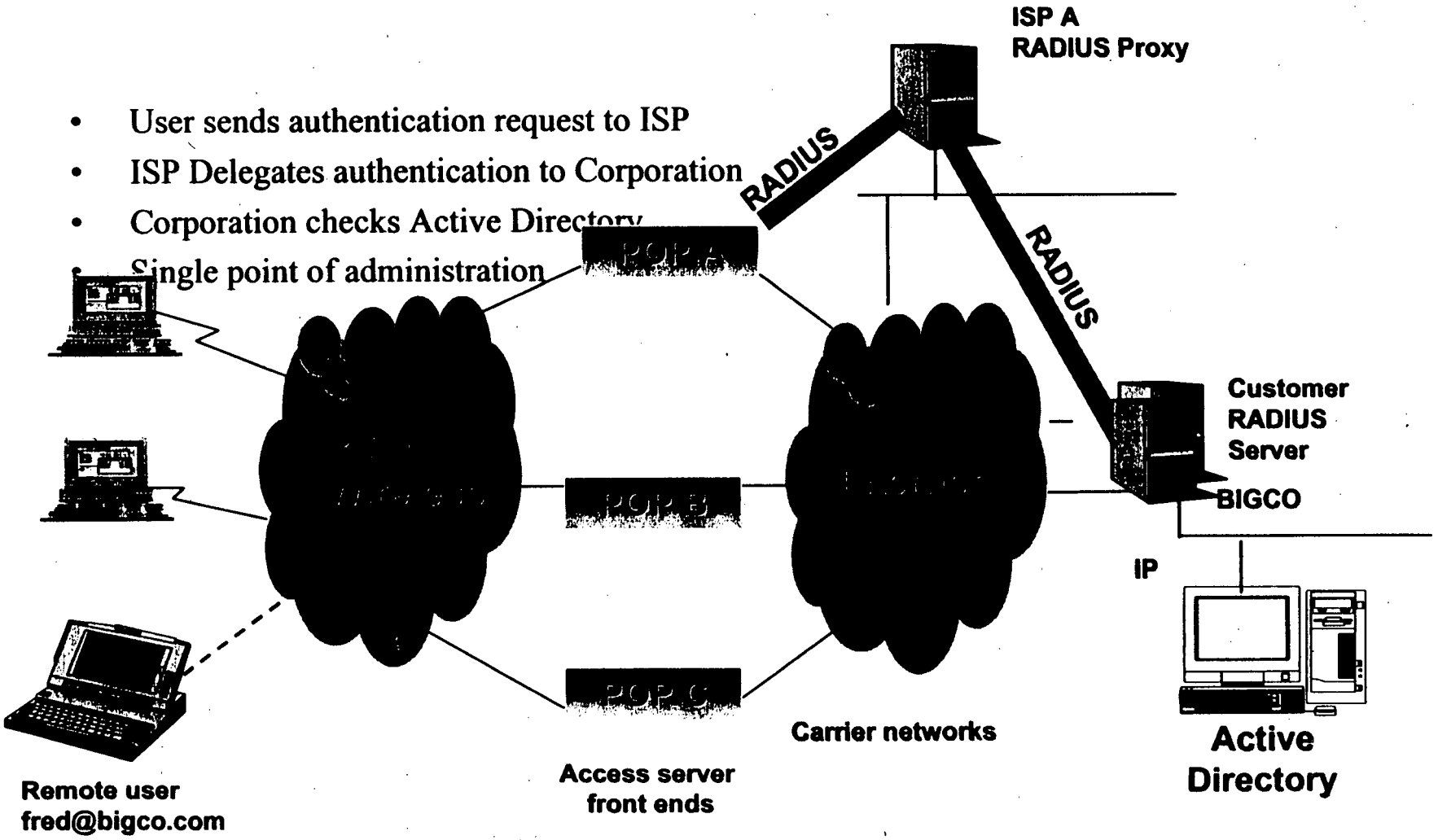
0956565 .0424400

802.1X On 802.11

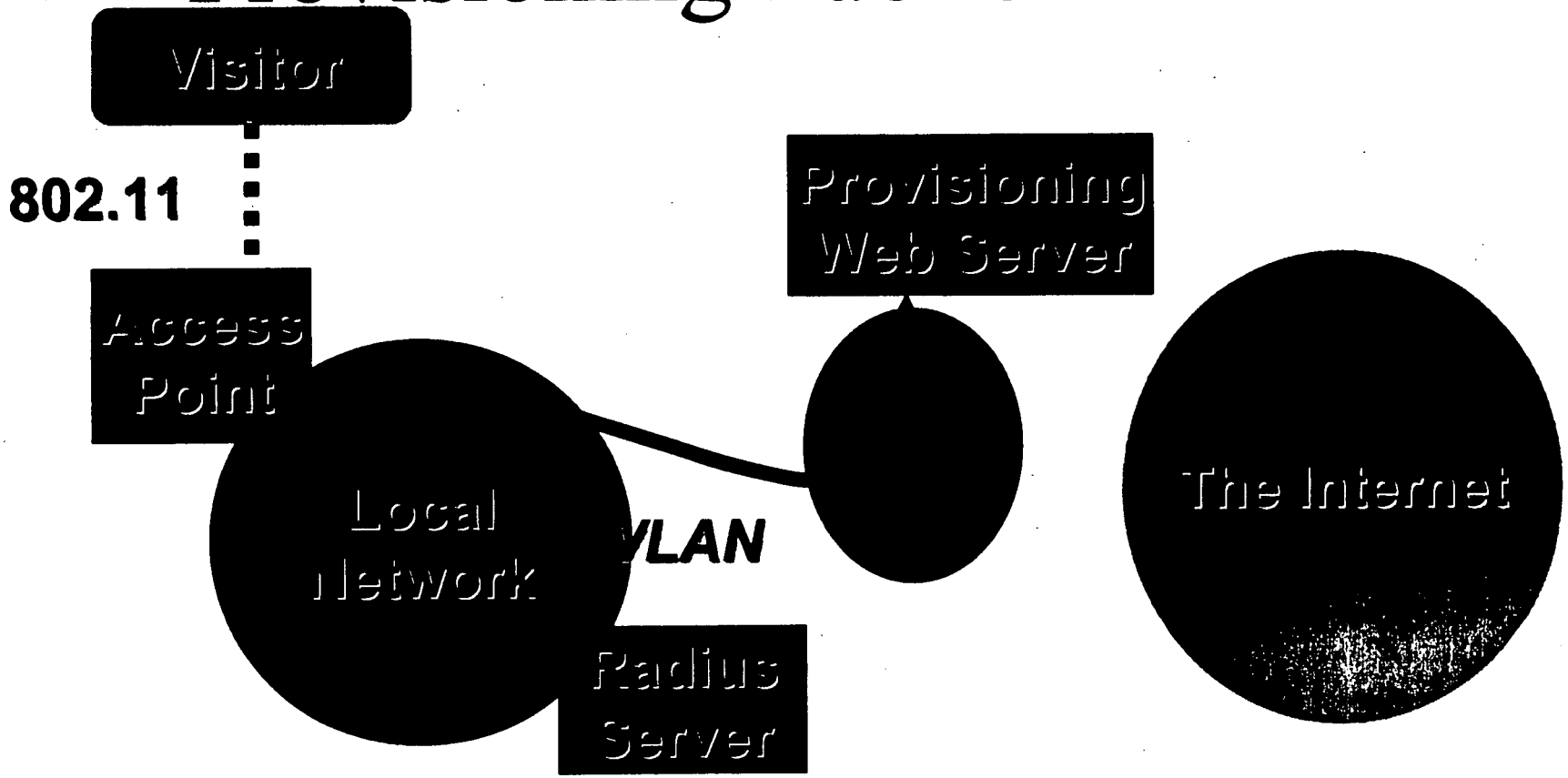


Outsourced Remote Access

- User sends authentication request to ISP
- ISP Delegates authentication to Corporation
- Corporation checks Active Directory
- Single point of administration



Provisioning Public Internet



Bluetooth Security

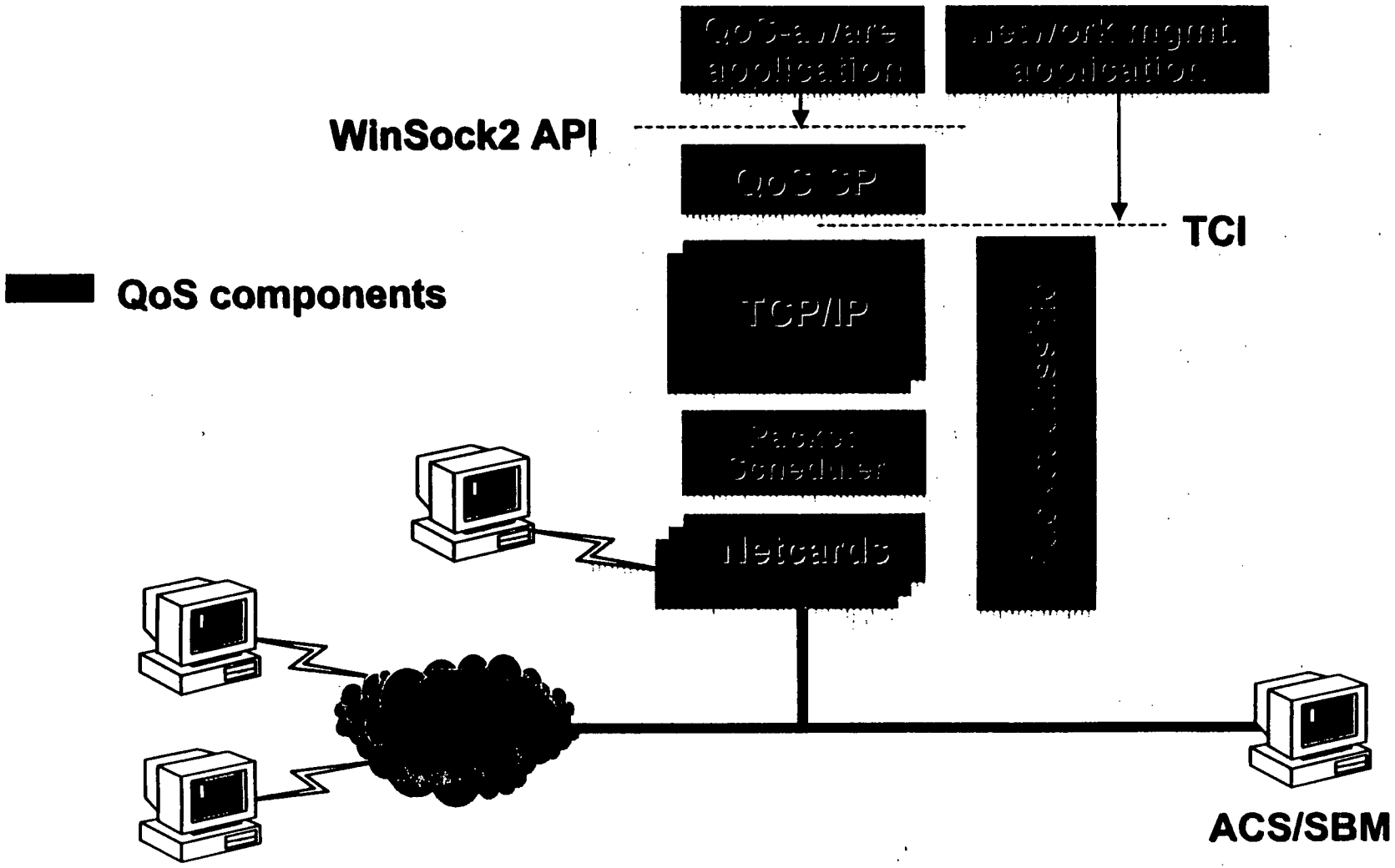
- To connect to a Bluetooth device requires its PIN
- PIN is per device not per service
 - Great for personal single function devices
 - E.g., protect cellphone from being dialed
- Problem for adhoc devices/applications
 - Require PIN for each device
 - Obtain access to all services on device
- Need security at a higher level and no PIN
 - Adhoc FTP user intervention required so why need a pin?
 - Adhoc PAN do not want a PIN otherwise cannot setup roaming PANs
 - Business card exchange should be push to a destination

GPRS Security

- GPRS uses GSM Authentication
- Authentication is between the mobile station and the network
 - Need authentication between PC and the Bluetooth mobile station
 - Bluetooth PIN

0956565 .0434.0

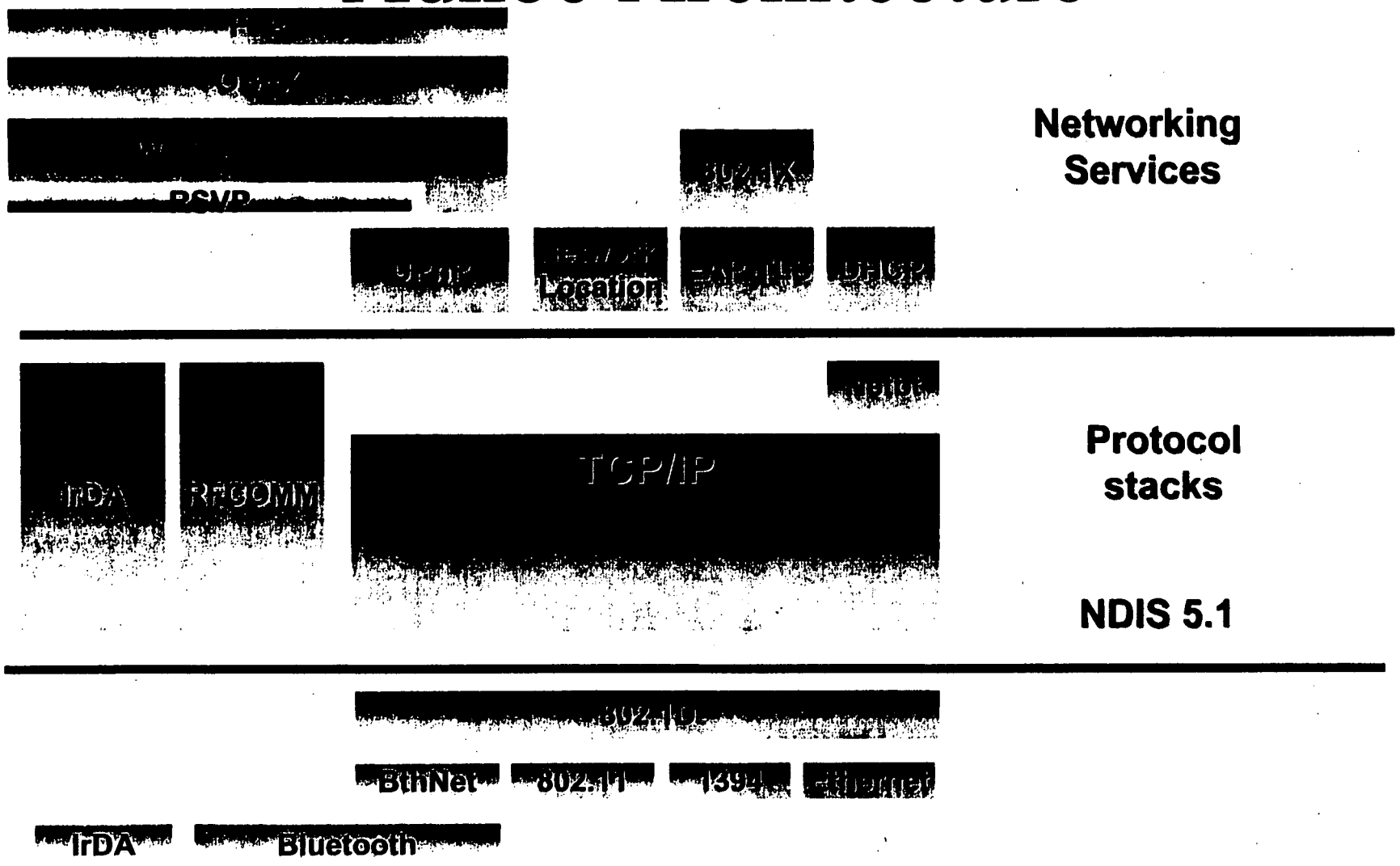
Microsoft® QoS Components



802.11 QoS

- 802.1p support
 - Priority tagging of Ethernet frames
- 802.11 NIC driver
 - Use NDIS priority field to prioritize access from client to wireless network
 - Add 802.1p header for wired network
- Access point prioritizes access from wired network to client based on 802.1p
- Subnetwork bandwidth manager in access point for admission control

Adhoc Architecture



No Network Infrastructure

- Address assignment
 - APIPA when no DHCP server
 - ICS contains DHCP server for adhoc home network
- Name Resolution
 - NetBT broadcast for adhoc name resolution
 - ICS contains DNS proxy and DDNS support for the adhoc home network
- Service Discovery Protocols
 - SSDP protocol enables UPnP discovery
 - SDP protocol enables Bluetooth wireless technology discovery
 - IrLAP protocol enables IrDA discovery

Temporary Networks

- Wireless allows for networks to be setup easily
- Interconnections not organized
 - Multiple interconnections to destinations
 - Loops in the network
- L2 Spanning tree
 - Self organizing networks
 - Removes loops

Ad Hoc Ethernet Networks

- Ethernet hubs
- Ethernet cross-over cables
- 1394
- Host to Host USB cables
- 802.11 can form adhoc mode
 - Automatically switch to adhoc mode when no access points in range
- Bluetooth wireless technology
- IrDA

0956565 .042400

IrDA/Bluetooth Architecture



IrDA Applications

- File transfer
 - Integrated into shell
- Image exchange from camera
- Dial-up networking via cellphone
- Printing
- Synchronization
 - ActiveSync®

Bluetooth Applications

- Subset of IrDA
- File transfer
 - Integrated into IrDA ftp transfer
- Dial-up Networking via cellphone
- IR and Bluetooth applications are tied to particular media
 - Do not inter-operate

Ad Hoc Applications

- UPnP is the integration point for ad hoc applications
- UPnP applications and services are available over any IP network
 - Ethernet, Wireless LAN, 1394, etc.

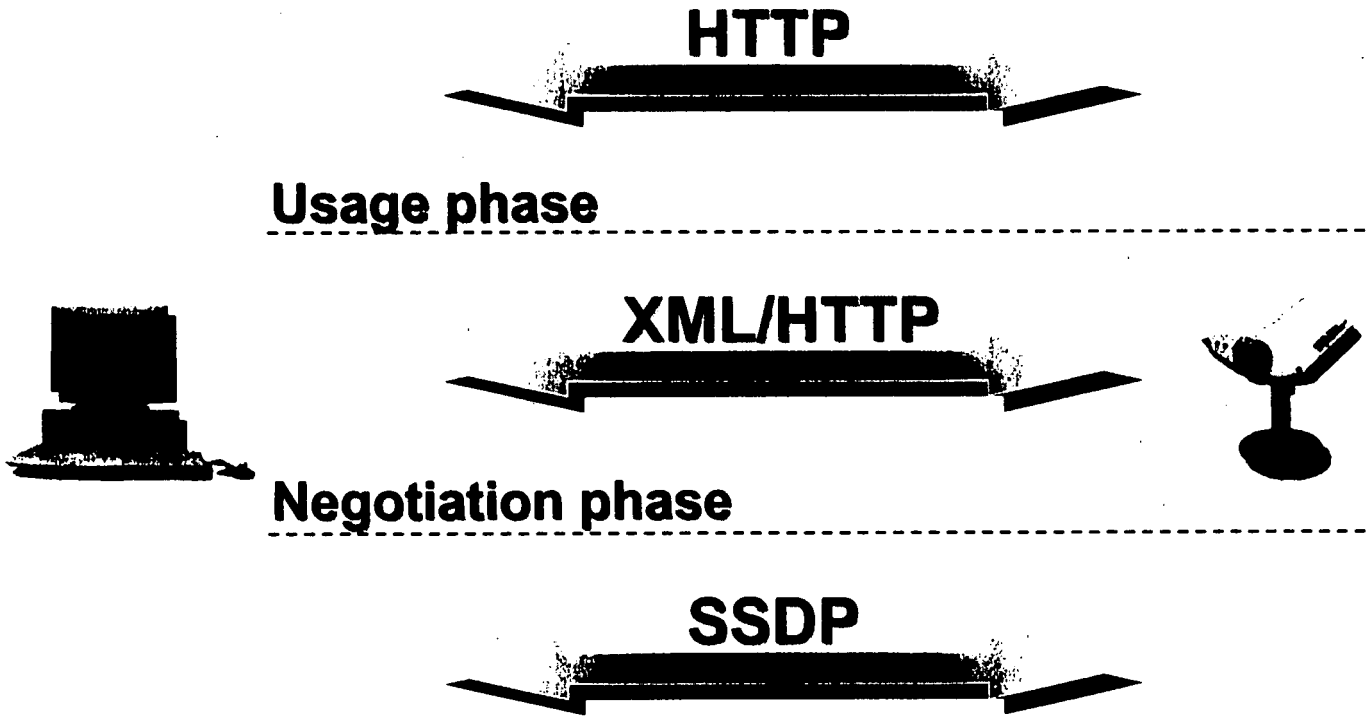
0956565.043400

UPnP Architecture Reference

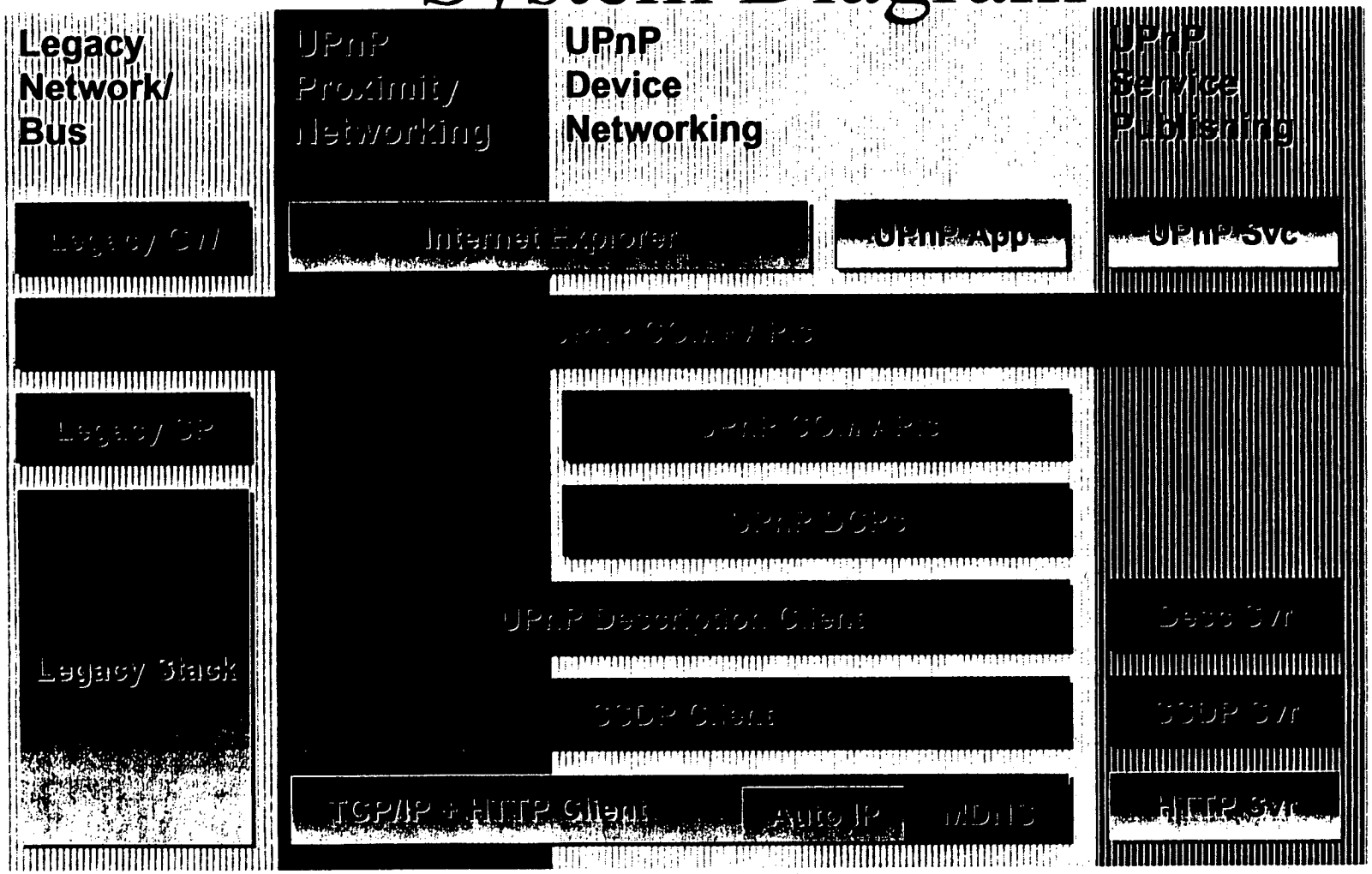


- Description/usage
 - Standardized protocols
 - Standardized XML descriptions
- Simple discovery
 - Locate devices/services on-the-fly
 - Standards-based

How It Works


HTTP
Usage phase
XML/HTTP
Negotiation phase
SSDP
Discovery phase

System Diagram



09556565 .042400

Wireless Performance

- TCP has many features optimized for wireless in Windows 2000
 - Improved RTT estimate
 - Improved window sizes
 - Fast retransmit
 - Select acknowledgement
 - Acknowledge packets
 - Improved time-out initiation
 - Very important for wireless losses
 - Cannot be used over the serial port
 - Use Remote NDIS
 - Over USB, IEEE 1394, Bluetooth wireless technology

WAP

- WAP was designed to remove some issues with TCP on long thin links
 - Remove 3 way handshake
 - Proposals to add data on the SYN and SYN-ACK
 - Reduces DOS protection
- Remove IP layer for some media
 - Not removed for GPRS
- Data compression
 - GPRS supports TCP/IP header and user data compression
 - Recommend GPRS systems support protocol header and user data compression
- WML is for small screens
 - E.g. a few lines

Summary – Wireless Is Here

- Bandwidth is growing
- Always connected wireless
- Enables new scenarios
 - Driving new applications
- Security a major issue with wireless
 - 802.1X allows integration into Windows user security system
- UPnP is the framework for adhoc applications

Call To Action

- Mobility
 - Mediasense is required for roaming support
 - Any wireless device must generate mediasense
- Implement 802.1X in network edge devices
 - Switches, access points, etc.
- Adhoc services and applications
 - Implement using UPnP
 - Do not limit your applications to a particular wireless media

For More Information

- Bluetooth wireless technology
 -
- IrDA
 -
- UPnP
 -
 -
- 802.11
 - QoS whitepaper
 - Security whitepaper
 - NIC requirements whitepaper

0956565 .0124100

For More Information

- RNDIS
 - WinHec driver talk
 -
- TCP/IP
 - Whitepaper
 -

For More Information

- IEEE 802.1X
 - –
- RADIUS
 - –
 - –
 - –
 - –
 - –
 - –
 - –
 - –
- EAP
 - –
 - –

1 Windows Wireless Architecture

Tim Moore
Lead Program Manager
Windows Networking
Microsoft Corporation

2 Agenda

- Wireless trends
 - WAN, LAN, PAN
- Scenarios
 - Adhoc, home, small business
 - Enterprise, ISP
- Wireless architecture
- Summary
- Call to action
- More information

3 Wireless Trends

- IP networks
- Always connected
- Increased bandwidth
- Convenience
- Moving from vertical market to horizontal markets
- Moving from proprietary to standards based
- Proliferation of smart devices
- New scenarios enabled
- Outsourcing
- Adhoc networks

4 Information Anytime, Anywhere
Connecting Everything

5 Data Speeds Today

Network	Speed*	Type of Data
American Mobile ARDIS	19.2/4.8 Kbps	Packet
BellSouth Wireless Data	8 Kbps	Packet
Cellular (Analog)	9.6/4.8 Kbps	Circuit-Switched
CDPD	19.2 Kbps	Packet
CDMA	14.4 Kbps	Circuit-Switched
Nextel	9.6 Kbps	Circuit-switched
GSM	9.6 Kbps	Circuit-Switched
Metricom	28.8 Kbps	Packet as Dial-up
TDMA**	One-Way SMS Only	None

*Typical data throughput speed is usually 50% of gross speed

**TDMA systems do not support data in the U.S. at this time

6 Wide-Area Wireless

7 Local-Area Wireless

8 Personal Area Wireless

9 Personal Area Wireless

- IrDA
 - Around since 1994
 - Available on every PC and lots of devices
 - >20 million existing IrDA devices
 - Camera, PDAs, cellphones, printers, keyboards
- Exploding market fueled by Bluetooth momentum
 - Bluetooth wireless technology is a defacto standard
 - Proliferation of smart devices, convenience of cable replacement, and new usage scenarios

10 Scenarios

- Adhoc
- Home
- Small business
- Enterprise
- ISP

11 Ad Hoc Networks

12 A Connected Home

13 A Connected Small Office

14 Enterprise

- Information at your fingertips
 - At meetings, in the office, on the road
 - Reliable, secure, multimedia LAN

15 Enterprise

- End-user can access the enterprise wireless network transparently over a secure connection
 - The network administrator has control over which users have access to the enterprise wireless LAN
- Enterprise can offer its employees access via ISPs which outsource their authentication to the enterprise
 - End-user has IP connectivity as soon as a CDPD or a GPRS modem is plugged in
 - Make cellphones an always connected Internet access point using GPRS

- End-User can use Netmeeting with wireless LAN, when out of range of LAN can continue to conference via IP connected cellphone

16 An ISP Connected Public Space
- Discovery of proximity services (flight schedules at airport, mall directories, ...)

17 ISP
- Need mixed technologies
 - Higher speed in hot spots, e.g., 802.11
- Need authentication so ISPs can charge
- Allow ISPs to integrate into existing Radius systems
 - Allows ISP roaming agreements
 - Same as outsource dial
- Need to be able to provision unauthenticated users

18 Wireless Architecture
- "Just works"
- Always connected
- Unified transport: IP
- Mobility
- Unified security model
- Adhoc
- QoS
- Performance

19 Wireless Architecture

20 Just Works
- No configuration
 - Especially when roaming
- CDPD
 - Configure Network Equipment Identifier
- 802.11
 - Configure network name and security keys
 - Per location
- Bluetooth wireless technology
 - Configure PIN numbers
 - Per device

21 802.11 Configuration
- Current 802.11 networks need to be configured with name of the network
 - Roaming between multiple networks difficult especially when security is implemented
- Automatically find a wireless network
 - If Access point is beaconing network name, attempt to use that network
 - If no infrastructure available then switch to adhoc mode

22 Always Connected
- Permanent IP connectivity should not use dial-up model

- A CDPD card should appears as a LAN card
- A GPRS, EDGE or 3G card or cellphone should appear as a LAN card
 - GPRS Terminal Type Recommendations
 - Cellphone needs to be Type A (voice and packet)
 - PC-Card can be Type C (packet only)
- Implement an NDIS driver or use Remote NDIS
 - Remote NDIS over Bluetooth connections

23 Remote NDIS

- Remote NDIS enables a bus-agnostic connection to devices that provide network access
- Remote NDIS is both a driver architecture and a command language

24 Unified Transport: IP

- All other media except Bluetooth wireless technology support always connected IP
- Ethernet over point-to-point Bluetooth connections
 - L2 bridge gives an adhoc L2 network
- Adhoc applications use UPnP over IP
- Expect large numbers of wireless connected devices
 - Move to IPv6 for addresses

25 Mobility

- Applications should not rely on having a network available all the time
 - Network connection can disappear at anytime
 - Applications should reconnect automatically if the network appears
- Clients hold state about the network
 - IP address
 - Routes
- Networks hold state about the client
 - Multicast distribution
 - Quality of service
 - Secure access
 - Machine name to IP address mapping
- How to detect when this state is out of date
- Applications also hold state about the network
 - TCP connections
 - E.g. Proxies, firewalls, etc.

26 Mobility

- Detect roaming
 - Mediasense detects working/non-working interfaces

- Mediasense detects interfaces changing their network connection
- IP address
 - Mediasense triggers a DHCP renew; If renew fails, DHCP gets a new IP address
 - DHCP updates DNS when an address changes
 - TCP/IP removes IP addresses if NIC not connected
 - Mobile IP allows IP address to stay the same when roaming

27 Mobile IP

- Mobile IP keeps the application IP address the same
 - IPv4 has two options
 - Change the network interface address to a local IP address
 - Use an ARP proxy to keep the same IP address
 - IPv6 only has first option
- Mobile IP Issues
 - How to route efficiently
 - IPv6 fixes this issue
 - Firewall traversal
 - Time to get a local address
 - Doesn't allow Voice over IP roaming
- Doesn't address any of the other issues with multicast, QoS, security, applications
- GPRS and 3G have network layer mobility
- No plans to support Mobile IP until IPv6

28 Mobility

- Multicast
 - Mediasense triggers IGMP refresh on roaming
- QoS
 - Mediasense triggers RSVP refresh on roaming
- Routes
 - Mediasense triggers router detect (IRDP) on roaming
 - Default interface metrics should depend interface speed
 - Routes to no longer existing interface addresses are removed
- Security
 - Mediasense triggers network authentication refresh
- Applications
 - Need to retry connections on connection failure and mediasense
 - Configurations based on network location

29 Network Location API

- Network location is a hint to the application of the network the machine is connected to
- Accessible via Winsock API

- Query for the connected networks
 - WSALookupServiceBegin
 - WSALookupServiceNext
 - WSALookupServiceEnd
- Request for notification when the connected networks changes
 - WSANSIoctl (,SIO_NSP_NOTIFY_CHANGE,...)
- Applications that need configuration per network should use this API
 - E.g., application proxies

30 Security
- Secure access to resources in the network
 - This is Windows login
- Secure transfer of data over the network
 - This is IPSec
 - Integrated into Windows credentials using PKI and Kerberos
- Secure access to the network
 - This is available for RAS and VPNs
 - Integrated into Windows credentials using PKI (EAP) and Radius
 - Supports roaming of identities
- No secure access to LAN networks
 - Very important for Wireless

31 Wireless Security Issues
- User loses wireless NIC, doesn't report it
 - Without user authentication, Intranet now accessible by attackers
 - Without centralized accounting and auditing, no means to detect unusual activity
 - Users who don't log on for periods of time
 - Users who transfer too much data, stay on too long
 - Multiple simultaneous logins
 - Logins from the "wrong" machine account
 - With global keys, large scale re-keying required

32 Wireless Deployment Issues
- User administration
 - Integration with existing user administration tools required (RADIUS, LDAP-based directories)
 - Create a Windows group for wireless
 - Any user or machine who is a member of the group has wireless access
 - Identification via User-Name easier to administer than MAC address identification
 - Usage accounting and auditing desirable
- Key management

- Static keys difficult to manage on clients, access points
- Proprietary key management solutions require separate user databases

33 802.1X Topology

34 IEEE 802.1X

- Enables interoperable user identification, centralized authentication, key management
 - Leverages existing standards: EAP, RADIUS
 - Compatible with existing roaming technologies, enabling use in hotels and public places
- User-based identification
 - Identification based on Network Access Identifier (RFC 2486) enables support for roaming access in public spaces (RFC 2607)
- Dynamic key management
- Centralized user administration
 - Support for RADIUS (RFC 2138, 2139) enables centralized authentication, authorization and accounting
 - RADIUS/EAP (draft-ietf-radius-ext-07.txt) enables encapsulation of EAP packets within RADIUS
- Supported on Ethernet, Token Ring and 802.11

35 Extensible Authentication Protocol

- Used by PPP for RAS and VPN
- Allows support for a number of authentication mechanisms
 - EAP designed to allow additional authentication methods to be deployed with no changes to the access point or client NIC
 - RFC 2284 includes support for password authentication (EAP-MD5), One-Time Passwords (OTP)
 - Windows 2000 supports smartcard authentication (RFC 2716) and Security Dynamics
- Radius server used for authentication and authorization
 - Integrated into Active Directory™ users and groups
 - Supports cross authentication for roaming

36 802.1X On 802.11

37 Outsourced Remote Access

- User sends authentication request to ISP
- ISP Delegates authentication to Corporation
- Corporation checks Active Directory
- Single point of administration

38 Provisioning Public Internet

39 Bluetooth Security

- To connect to a Bluetooth device requires its PIN
- PIN is per device not per service
 - Great for personal single function devices
 - E.g., protect cellphone from being dialed

- Problem for adhoc devices/applications
 - Require PIN for each device
 - Obtain access to all services on device
- Need security at a higher level and no PIN
 - Adhoc FTP user intervention required so why need a pin?
 - Adhoc PAN do not want a PIN otherwise cannot setup roaming PANs
 - Business card exchange should be push to a destination

40 GPRS Security
- GPRS uses GSM Authentication
- Authentication is between the mobile station and the network
 - Need authentication between PC and the Bluetooth mobile station
 - Bluetooth PIN

41 Microsoft® QoS Components

42 802.11 QoS
- 802.1p support
 - Priority tagging of Ethernet frames
- 802.11 NIC driver
 - Use NDIS priority field to prioritize access from client to wireless network
 - Add 802.1p header for wired network
- Access point prioritizes access from wired network to client based on 802.1p
- Subnetwork bandwidth manager in access point for admission control

43 Adhoc Architecture

44 No Network Infrastructure
- Address assignment
 - APIPA when no DHCP server
 - ICS contains DHCP server for adhoc home network
- Name Resolution
 - NetBT broadcast for adhoc name resolution
 - ICS contains DNS proxy and DDNS support for the adhoc home network
- Service Discovery Protocols
 - SSDP protocol enables UPnP discovery
 - SDP protocol enables Bluetooth wireless technology discovery
 - IrLAP protocol enables IrDA discovery

45 Temporary Networks
- Wireless allows for networks to be setup easily
- Interconnections not organized
 - Multiple interconnections to destinations
 - Loops in the network
- L2 Spanning tree
 - Self organizing networks

- Removes loops

46 Ad Hoc Ethernet Networks
- Ethernet hubs
- Ethernet cross-over cables
- 1394
- Host to Host USB cables
- 802.11 can form adhoc mode
 - Automatically switch to adhoc mode when no access points in range
- Bluetooth wireless technology
- IrDA

47 IrDA/Bluetooth Architecture

48 IrDA Applications
- File transfer
 - Integrated into shell
- Image exchange from camera
- Dial-up networking via cellphone
- Printing
- Synchronization
 - ActiveSync®

49 Bluetooth Applications
- Subset of IrDA
- File transfer
 - Integrated into IrDA ftp transfer
- Dial-up Networking via cellphone
- IR and Bluetooth applications are tied to particular media
 - Do not inter-operate

50 Ad Hoc Applications
- UPnP is the integration point for ad hoc applications
- UPnP applications and services are available over any IP network
 - Ethernet, Wireless LAN, 1394, etc.

51 UPnP Architecture Reference
- Description/usage
 - Standardized protocols
 - Standardized XML descriptions
- Simple discovery
 - Locate devices/services on-the-fly
 - Standards-based

52 How It Works

53 System Diagram

54 Wireless Performance

- TCP has many features optimized for wireless in Windows 2000
 - Improved RTT estimate
 - Improved window sizes
 - Fast retransmit
 - Select acknowledgement
 - Acknowledge packets
 - Improved time-out initiation
 - Very important for wireless losses
 - Cannot be used over the serial port
 - Use Remote NDIS
 - Over USB, IEEE 1394, Bluetooth wireless technology

55 WAP

- WAP was designed to remove some issues with TCP on long thin links
 - Remove 3 way handshake
 - Proposals to add data on the SYN and SYN-ACK
 - Reduces DOS protection
- Remove IP layer for some media
 - Not removed for GPRS
- Data compression
 - GPRS supports TCP/IP header and user data compression
 - Recommend GPRS systems support protocol header and user data compression
- WML is for small screens
 - E.g., a few lines

56 Summary – Wireless Is Here

- Bandwidth is growing
- Always connected wireless
- Enables new scenarios
 - Driving new applications
- Security a major issue with wireless
 - 802.1X allows integration into Windows user security system
- UPnP is the framework for adhoc applications

57 Call To Action

- Mobility
 - Mediasense is required for roaming support
 - Any wireless device must generate mediasense
- Implement 802.1X in network edge devices
 - Switches, access points, etc.
- Adhoc services and applications

- Implement using UPnP
- Do not limit your applications to a particular wireless media

58 For More Information

- Bluetooth wireless technology
 - http://www.bluetooth.com
- IrDA
 - http://www.irda.org
- UPnP
 - http://www.upnp.org
 - http://www.microsoft.com/hwdev/upnp
- 802.11
 - QoS whitepaper
 - Security whitepaper
 - NIC requirements whitepaper

59 For More Information

- RNDIS
 - WinHec driver talk
 - http://www.microsoft.com/hwdev/network
- TCP/IP
 - Whitepaper
 - http://www.microsoft.com/windows2000/library/howitworks/communications/networkbasics/tcpip_implement.asp

60 For More Information

- IEEE 802.1X
 - http://grouper.ieee.org/groups/802/1/pages/802.1x.html
- RADIUS
 - http://www.ietf.org/rfc/rfc2138.txt
 - http://www.ietf.org/rfc/rfc2139.txt
 - http://www.ietf.org/rfc/rfc2548.txt
 - http://www.ietf.org/internet-drafts/draft-ietf-radius-radius-v2-06.txt
 - http://www.ietf.org/internet-drafts/draft-ietf-radius-accounting-v2-05.txt
 - http://www.ietf.org/internet-drafts/draft-ietf-radius-ext-07.txt
 - http://www.ietf.org/internet-drafts/draft-ietf-radius-tunnel-auth-09.txt
 - http://www.ietf.org/internet-drafts/draft-ietf-radius-tunnel-acct-05.txt
- EAP
 - http://www.ietf.org/rfc/rfc2284.txt
 - http://www.ietf.org/rfc/rfc2716.txt

00h2h0" 9959560

APPENDIX C

Bluetooth Architecture Overview

James Kardach
Principal Engineer/Bluetooth SIG Chairman
Mobile Computer Group
Intel Corporation

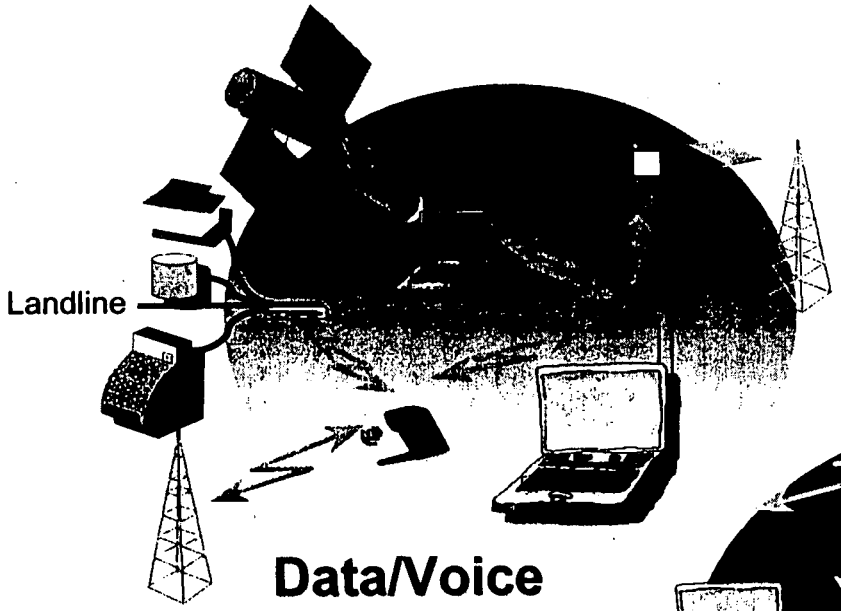


"The Bluetooth Specification is still preliminary; All information regarding Bluetooth is subject to change without notice"

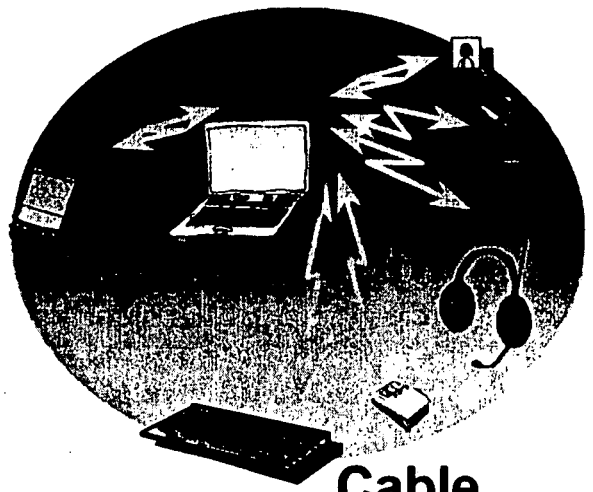
Agenda

- Bluetooth SIG update
- Bluetooth architectural overview
- Summary/call to action

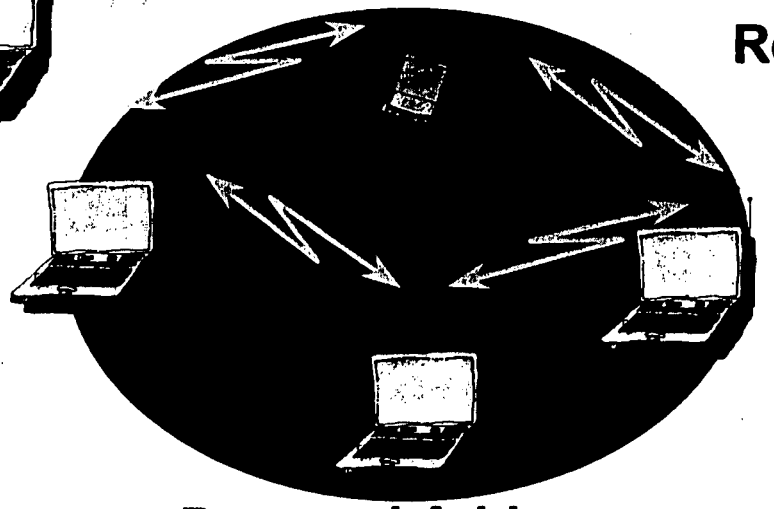
What Does Bluetooth Wireless Technology Do For You?



Data/Voice Access Points



Cable Replacement

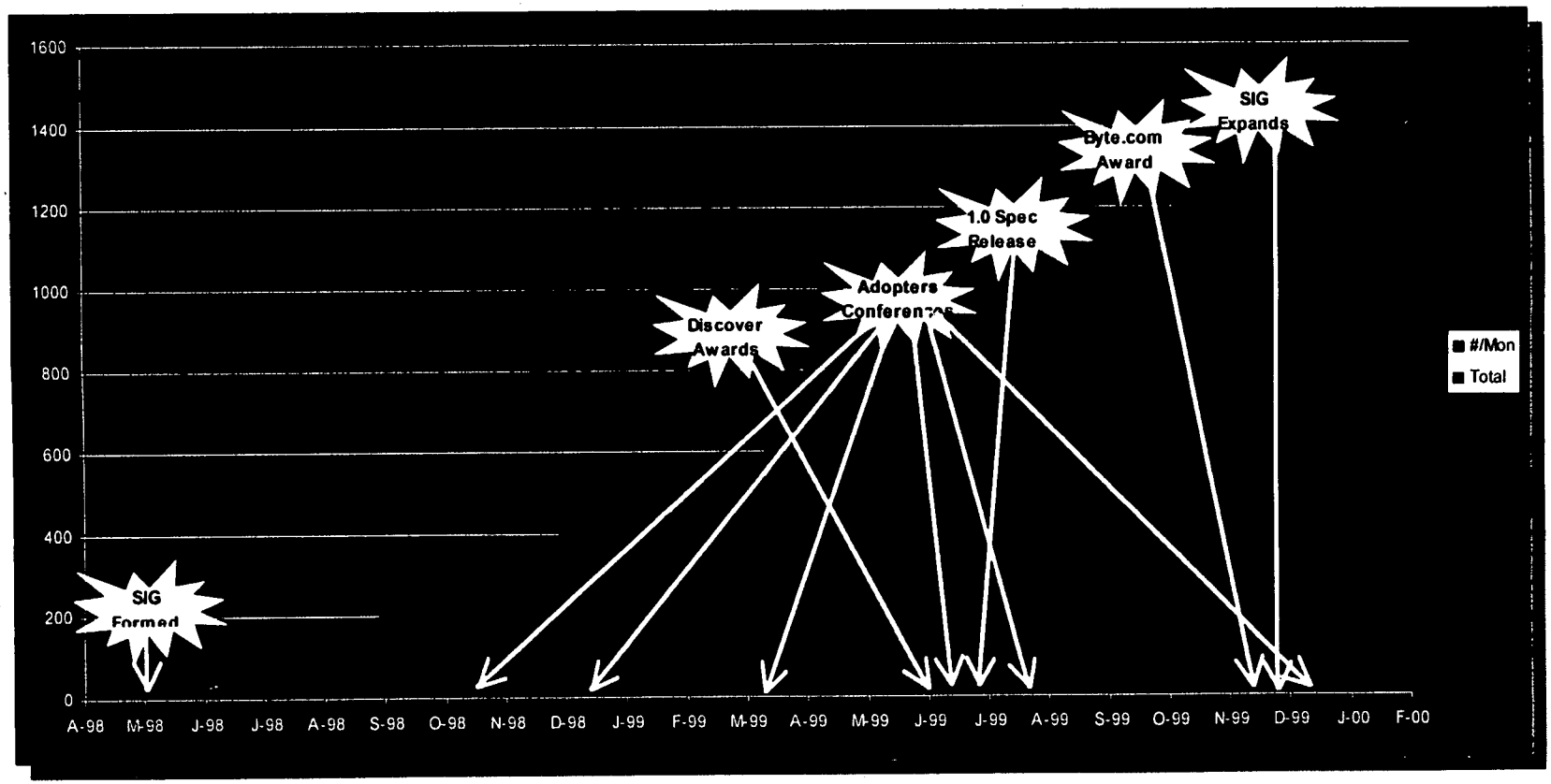


Personal Ad-hoc Networks

Program Update

- Final specification published Monday 7/26/99
 - Core technology specs and Profile requirements
 - Result of work from ~200 engineers
- Bluetooth membership exceeds 1,600 companies!
 - Bluetooth wireless technology becoming the choice for wireless connectivity
 - Full list of member companies on Web site
- Bluetooth program on track for products available in 2000
 - Products available this year
 - Next step is qualification program
 - SIG now focusing on ensuring product interoperability
- Bluetooth qualification program started
- Bluetooth wireless technology is the basis for the IEEE 802.15.1 standard
- Bluetooth SIG has expanded
 - New contracts and membership types

Bluetooth Momentum



- SIG is still growing

The SIG Formally Known As Bluetooth ;)

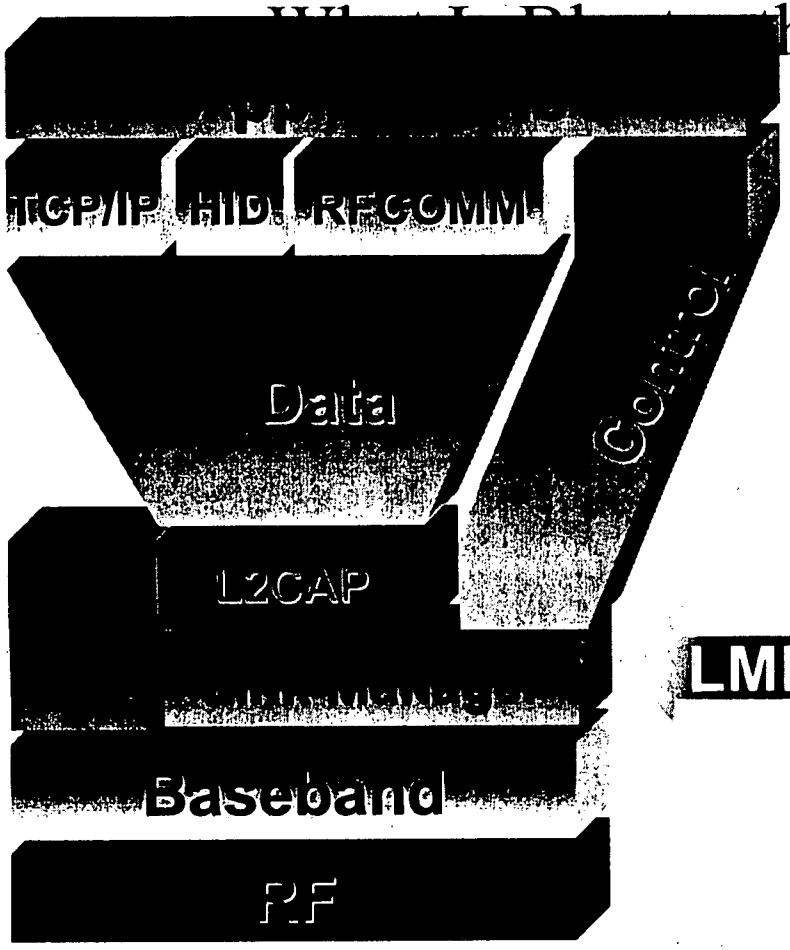
- New Contracts
 - Adopter/Early Adopter = Early Adopter
 - Early Adopter Contract
 - Early Adopter in working group = Associate
 - Early Adopter Contract, Associate Amendment
- Open IP license to Bluetooth wireless technology
 - Original "Foundation Specifications"
 - New technology in and around the 12 specification working groups
- Only need to sign 1 contract to use any Bluetooth wireless technology (the new one)

Confidential

Future Directions For Bluetooth Profiles

- Second generation radio
- Personal area networking
- In and around the car
- "Wake-up"
- Human Interface Devices (HID)
- Audio/visual
- ISM interference/interoperability
- Printing
- Still image
- Extended Service Discovery protocols
- Local positioning
- UDI

h Wireless Technology?



- A hardware description
- An application framework

Application Framework and Support

Host Controller Interface

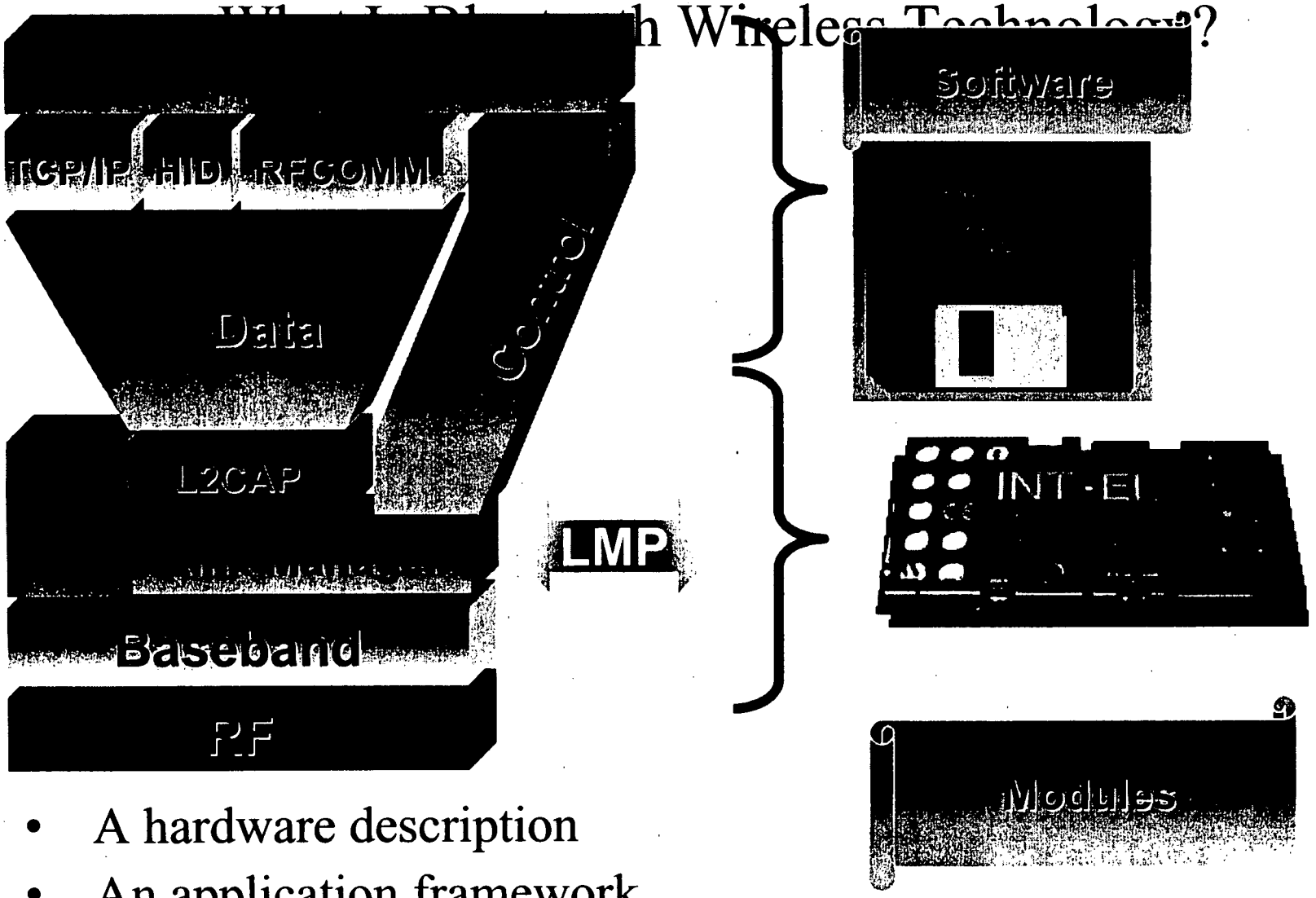
Link Manager and L2CAP

Radio and Baseband

Latest Version on Bluetooth Website:
www.Bluetooth.com

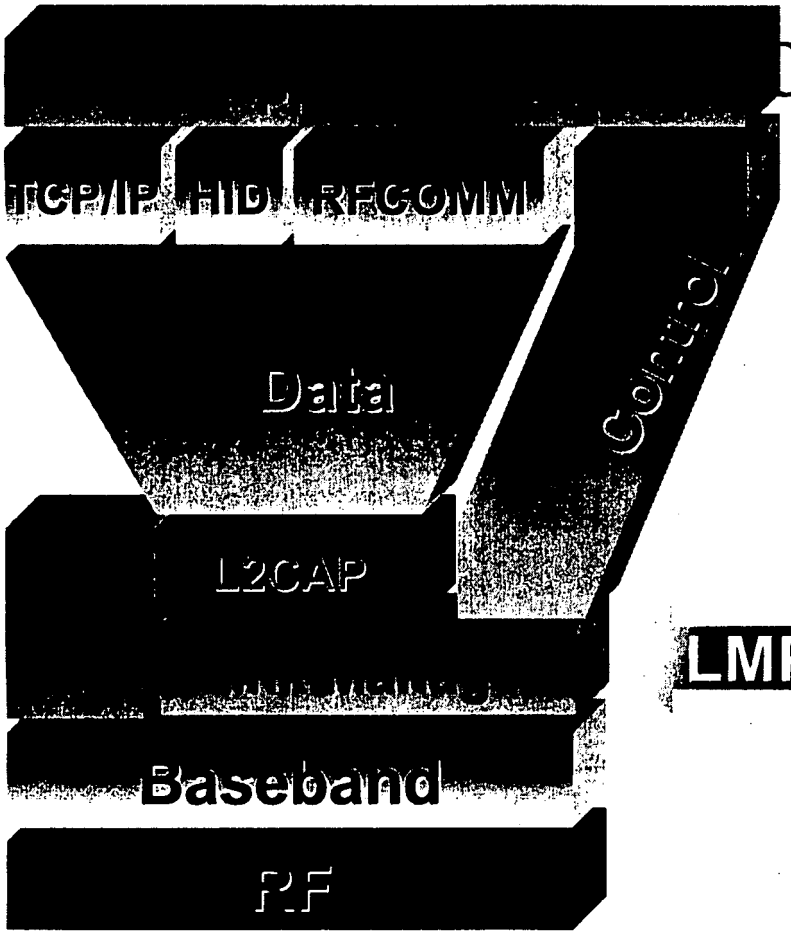
0h2h0' 9999960

h Wireless Technology?



- A hardware description
- An application framework

Specification



Application Framework Certification

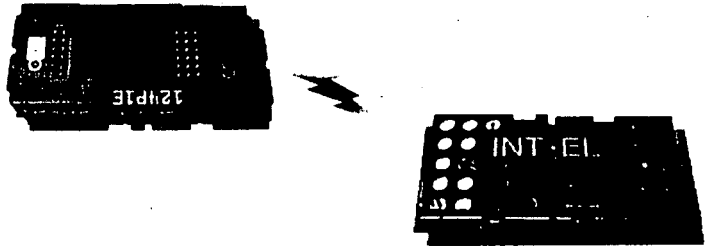
vCard	IrOBEX	BT.OBEX	BT.vCard
UDP PPP IrOBEX	PPP RFCOMM RFCOMM	BT.PPP BT.TS0710 BT.TS0710	BT.UDP BT.PPP BT.OBEX
Still Images	HID	BT.HID	BT.SImg
Audio Ctrl	L²CAP	BT.L²CAP-A	BT.AudioCtrl
TCP/IP	L²CAP	BT.L²CAP-D	BT.TCP/IP

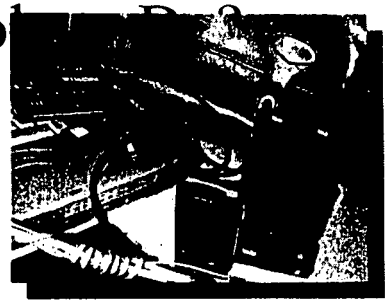
Basic Layer Certification

L²CAP	LM	BT.LM-A	BT.LM-D	BT.L²CAP-A	BT.L²CAP-D
BB RF	RF Air	BT.RF	BT.RF	BT.BB-A	BT.BB-D

- Bluetooth devices will be tested against the specification
- Bluetooth Qualified Test Facilities (BQTF)

What Does Bluetooth Wireless Techno





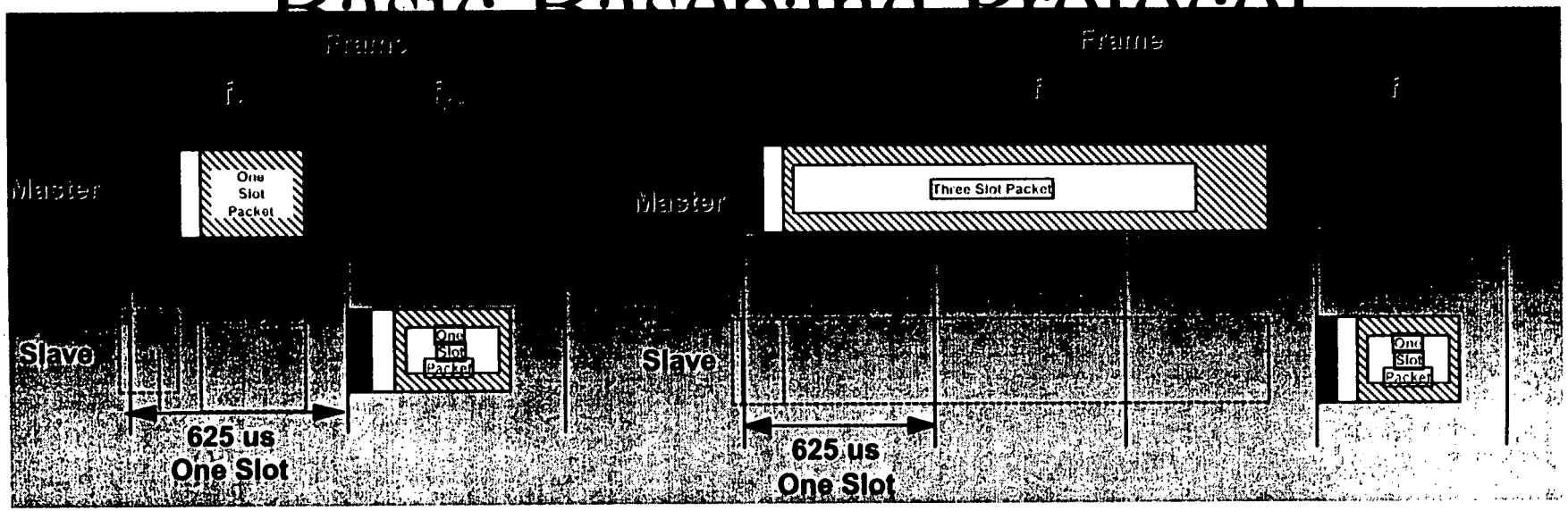
Topology	Supports up to 7 simultaneous links	Each link requires another cable
Flexibility	Goes through walls, bodies, cloths...	Line of sight or modified environment
Data rate	1 MSPS, 720 Kops	Varies with use and cost
Power	0.1 watts active power	0.05 watts active power or higher
Size/Weight	25 mm x 13 mm x 2 mm, several grams	Size is equal to range. Typically 1 -2 meters. Weight varies with length (ounces to pounds)
Cost	Long-term $5 per endpoint	~ $8 -$100/meter (end user cost)
Range	10 meters or less Up to 100 meters with PA.	Range equal to size. Typically 1 -2 meters
Universal	Intended to work anywhere in the world	Cables vary with local customs
Security	Very, link layer security, SS radio	Secure (its a cable)

- Cable Replacement

Bluetooth RF Specifications

- Specified for low cost, single chip implementation
 - Noise floor margin for substrate noise and low current LNA
 - Linearity set by near-far problem
 - In-band image allows low-cost low IF
 - VCO phase noise enables integrated VCO
 - TX-RX turn around time enables single synthesizer
 - 2.4 ISM band chosen for global use and process capabilities
- Sensitivity traded for low-cost integration of transceiver and baseband

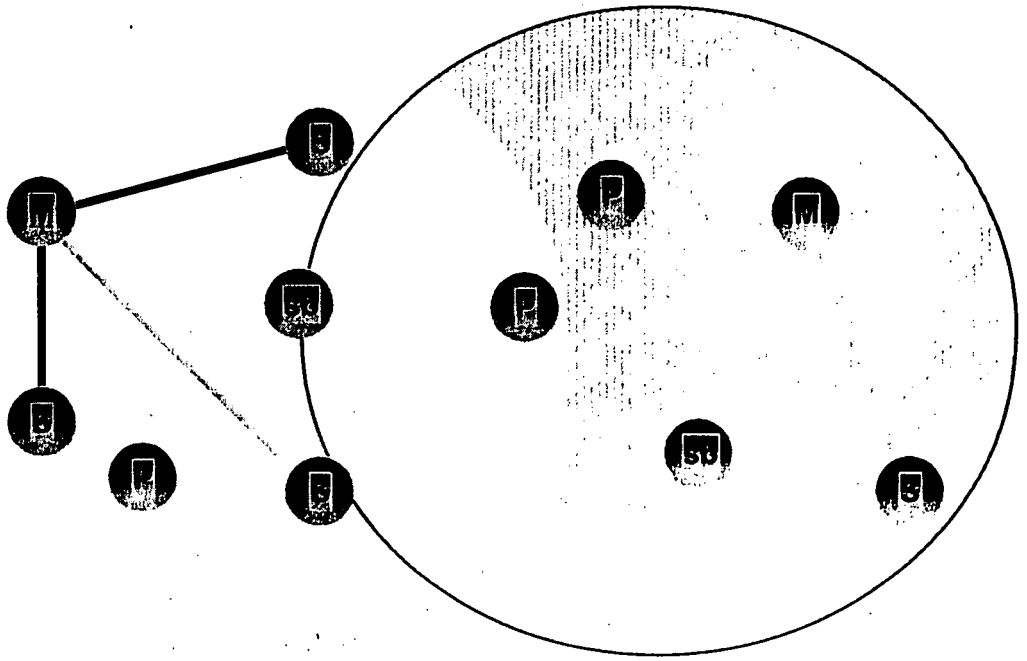
Basic Baseband Protocol



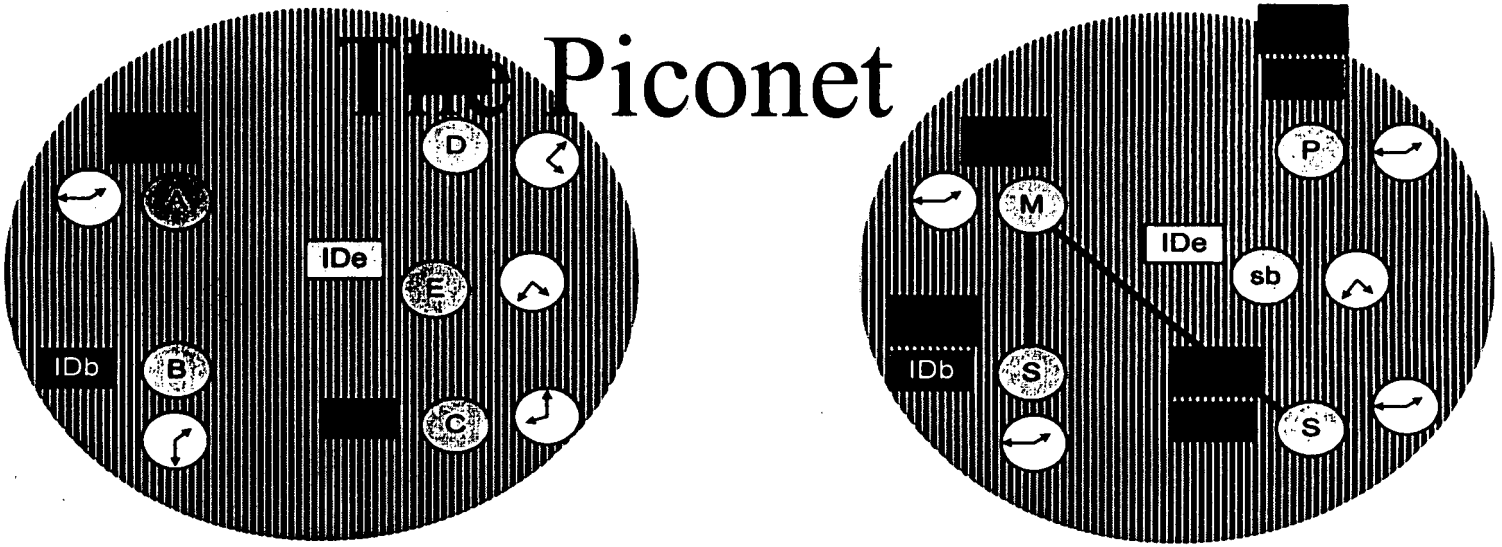
- Spread spectrum frequency hopping radio
 - 79/23 one MHz channels
 - Hops every packet
 - Packets are 1, 3, or 5 slots long
 - Frame consists of two packets
 - Transmit followed by receive
 - Nominally hops at 1600 times a second (1 slot packets)

Network Topology

- Radio Designation
 - Connected radios can be master or slave
 - Radios are symmetric (same radio can be master or slave)
- Piconet
 - Master can connect to 7 simultaneous or 200+ active slaves per piconet
 - Each piconet has maximum capacity (1 MSPS)
 - Unique hopping pattern/ID
- Scatternet
 - High capacity system
 - Minimal impact with up to 10 piconets within range
 - Radios can share piconets!



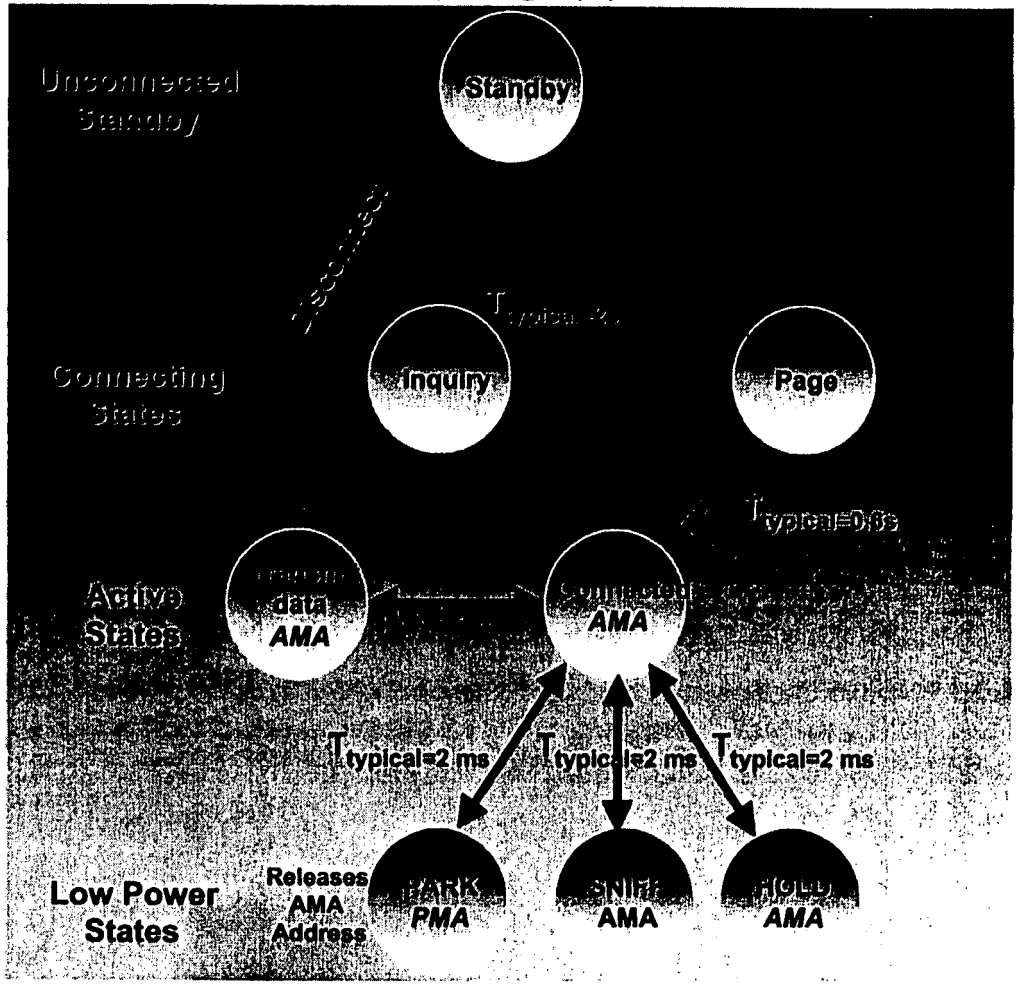
The Piconet



- All devices in a piconet hop together
 - In forming a piconet, master gives slaves its clock and device ID
 - Hopping pattern determined by device ID (48-bit)
 - Phase in hopping pattern determined by Clock
- Non-piconet devices are in standby
- Piconet Addressing
 - Active Member Address (AMA, 3-bits)
 - Parked Member Address (PMA, 8-bits)

Functional Overview

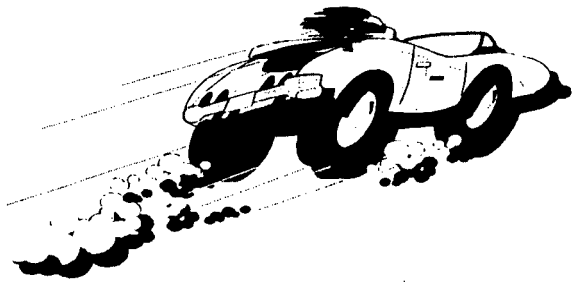
- Standby
 - Waiting to join a piconet
- Inquire
 - Ask about radios to connect to
- Page
 - Connect to a specific radio
- Connected
 - Actively on a piconet (master or slave)
- Park/Hold/Sniff
 - Low Power connected states



Packet Types/Data Rates

Packet Types			
SEGMENT	TYPE	SCO link	ASL link
1	0000 0001 0010 0011	NULL POLL FHS DM1	NULL POLL FHS DM1
2	0100 0101 0110 0111 1000 1001	HV1 HV2 HV3 DV	DH1 AUX1
3	1010 1011 1100 1101		DM3 DH3
4	1110 1111		DM5 DH5

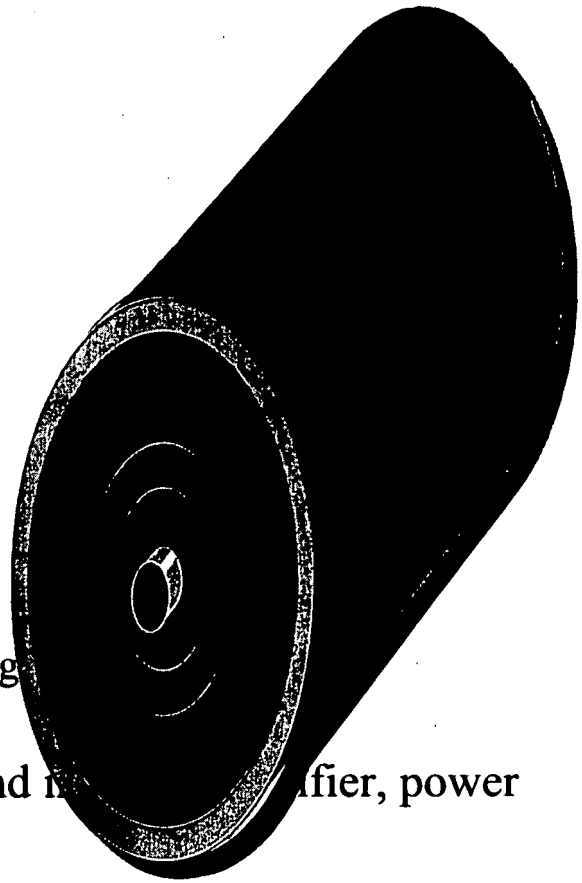
Data Rates (Kbps)			
TYPE	symmetric	asymmetric	
DM1	108.8	108.8	108.8
DH1	172.8	172.8	172.8
DM3	256.0	384.0	54.4
DH3	384.0	576.0	86.4
DM5	286.7	477.8	36.3
DH5	432.6	721.0	57.6



- ASL –Packet like behavior
- SCO – Circuit like behavior

Mobile = Battery Life

- Low power consumption*
 - Standby current < 0.3 mA
 - Þ 3 months
 - Voice mode 8-30 mA
 - Þ 75 hours
 - Data mode average 5 mA
 - (0.3-30mA, 20 kbit/s, 25%)
 - Þ 120 hours
- Low Power Architecture
 - Programmable data length (else radio sleeps)
 - Hold and Park modes 60 µA
 - Devices connected but not participating
 - Hold retains AMA address, Park releases AMA, g
 - Device can participate within 2 ms
 - *Estimates calculated with 600 mAh battery and i ifier, power will vary with implementation



Error Handling

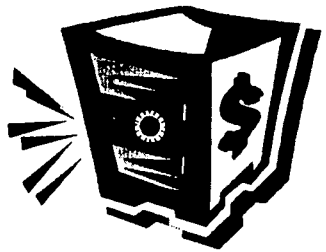


- Forward-error correction (FEC)
 - Headers are protected with 1/3 rate FEC and HEC
 - Payloads may be FEC protected
 - 1/3 rate: simple bit repetition (SCO packets only)
 - 2/3 rate: (10,15) shortened Hamming code
 - 3/3 rate: no FEC
- ARQ (ACL packets only)
 - 16-bit CRC (CRC-CCITT) and 1-bit ACK/NACK
 - 1-bit sequence number

0n2h0" 999960

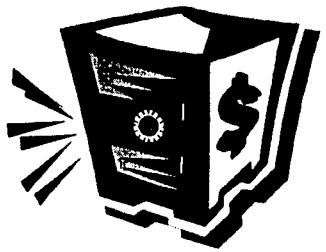
Bluetooth Security Features

- Fast Frequency Hopping (79 channels)
- Low Transmit Power (range <= 10m)
- Authentication of remote device
 - Based on link key (128 Bit)
 - May be performed in both directions
- Encryption of payload data
 - Stream cipher algorithm (≤ 128 Bit)
 - Affects all traffic on a link
- Initialization
 - PIN entry by user



Application Level Security

- Builds on-top of link-level security
 - Creates trusted device groups
- Security levels for services
 - Authorization required
 - Authentication required
 - Encryption required
- Different or higher security requirements could be added:
 - Personal authentication
 - Higher security level
 - Public key



Bluetooth Wireless Technology Is Global

- One version for the world
 - Architecture compliant with global emission rules (2.4 GHz ISM band)
 - Working through FCC, EC, MPT for spectrum, and power harmonization
 - Architecture compliant and safe for use on airlines
 - Working with FAA, JAA, FCC, airplane manufacturers, and airlines
 - Reviewing security architecture with affected countries

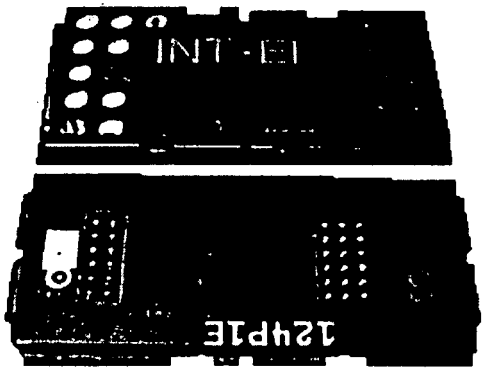


Bluetooth Radio Modules

- Complete radio on a module
 - Designed to meet "Limited Module Compliance" (LMA) requirements
 - Pre-certified to meet global regulatory requirements
 - Allows devices assembled with modules to be "self-certified"
 - USB Interface
 - Solder-ball connections
 - External Antennae



25 mm dia

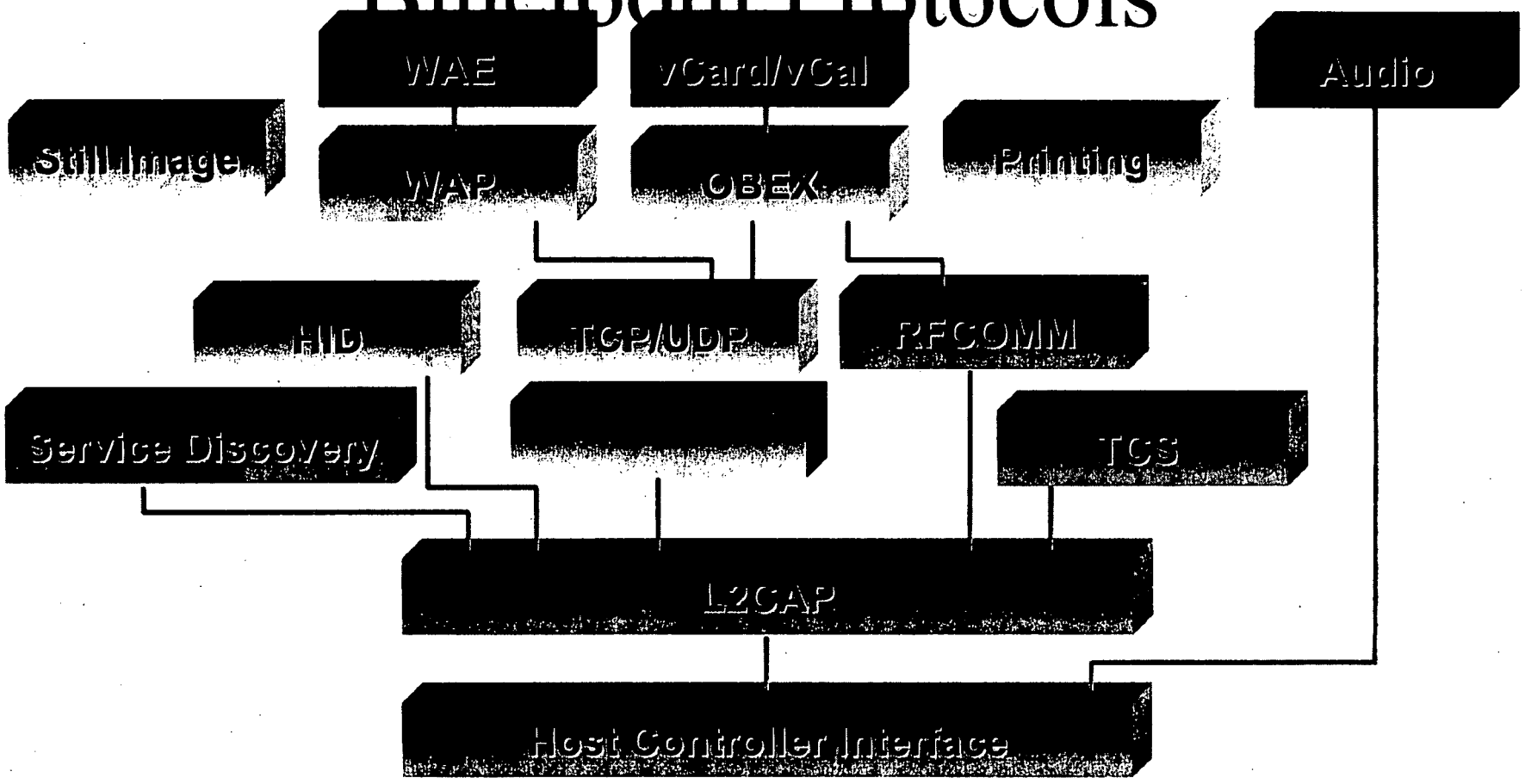


17x33mm

Compact FLASH Card

36x43mm

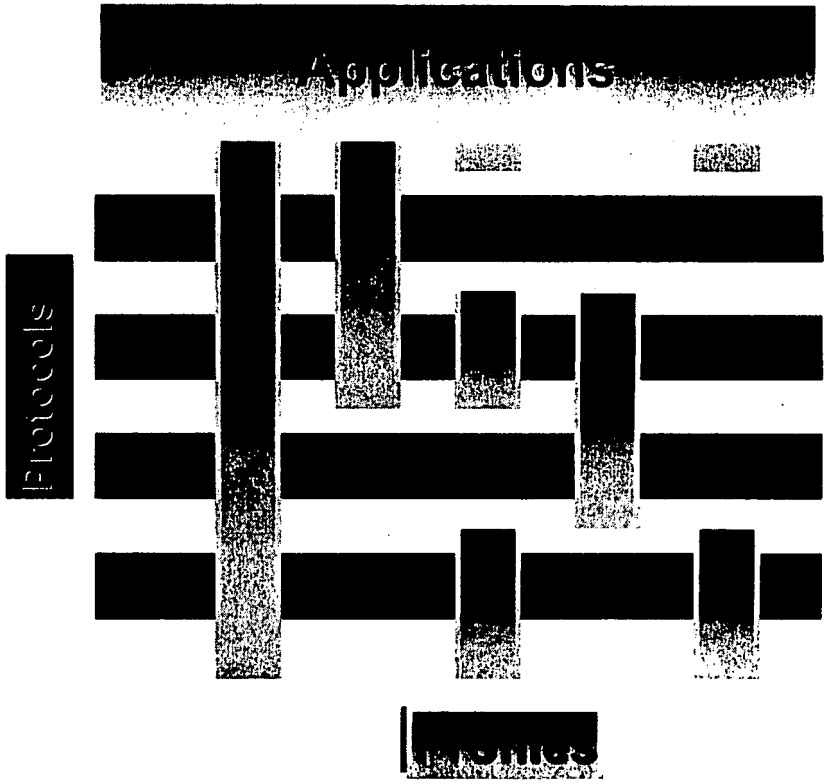
Bluetooth Protocols



- Bluetooth Specific
- Reused Spec
- Modified

Interoperability And Profiles

- Represents default solution for usage model
- Vertical slice through the protocol stack
- Basis for interoperability and logo requirements
- Each Bluetooth device supports one or more profiles



Summary

- Bluetooth wireless technology is a global, RF-based (ISM: 2.4GHz band), short-range, connectivity technology and solution for portable, personal devices
 - It is not just a radio
 - Create piconets on-the-fly (approximately 1Mbps)
 - Piconets may overlap in time and space for high aggregate bandwidth
- The Bluetooth spec comprises
 - A hardware and software protocol specification
 - Usage case scenario profiles and interoperability requirements
- To learn more: http://www.bluetooth.com

Call To Action

- Join the SIG if you haven't already
 - Help advance Bluetooth functionality by supporting the working groups committees
 - Got a new usage model? Submit a request
- Learn how Bluetooth wireless technology works NOW!
 - See Microsoft's presentation on Bluetooth wireless technology
 - Big conference in Monte Carlo - check it out!
 - More information:
- Implement Bluetooth software and hardware in your products and systems
 - Insure interoperability via Un-plugfests
- Help support native operating system development
 - Provide test hardware to Microsoft

1 Bluetooth Architecture Overview

James Kardach
Principal Engineer/Bluetooth SIG Chairman Mobile Computer Group
Intel Corporation

2 Agenda

- Bluetooth SIG update
- Bluetooth architectural overview
- Summary/call to action

3 What Does Bluetooth Wireless Technology Do For You?

4 Program Update

- Final specification published Monday 7/26/99
 - Core technology specs and Profile requirements
 - Result of work from ~200 engineers
- Bluetooth membership exceeds 1,600 companies!
 - Bluetooth wireless technology becoming the choice for wireless connectivity
 - Full list of member companies on Web site www.bluetooth.com
- Bluetooth program on track for products available in 2000
 - Products available this year
 - Next step is qualification program
 - SIG now focusing on ensuring product interoperability
- Bluetooth qualification program started
- Bluetooth wireless technology is the basis for the IEEE 802.15.1 standard
- Bluetooth SIG has expanded
 - New contracts and membership types

5 Bluetooth Momentum

- SIG is still growing

6 The SIG Formally Known As Bluetooth ;)

- New Contracts
 - Adopter/Early Adopter = Early Adopter
 - Early Adopter Contract
 - Early Adopter in working group = Associate
 - Early Adopter Contract, Associate Amendment
- Open IP license to Bluetooth wireless technology
 - Original "Foundation Specifications"
 - New technology in and around the 12 specification working groups
- Only need to sign 1 contract to use any Bluetooth wireless technology (the new one)

7 Future Directions For Bluetooth Profiles

- Second generation radio

Bluetooth - Confidential

- Personal area networking
- In and around the car
- "Wake-up"
- Human Interface Devices (HID)
- Audio/visual
- ISM interference/interoperability
- Printing
- Still image
- Extended Service Discovery protocols
- Local positioning
- UDI

8 What Is Bluetooth Wireless Technology?
- A hardware description
- An application framework

9 What Is Bluetooth Wireless Technology?
- A hardware description
- An application framework

10 Testing To Specification
- Bluetooth devices will be tested against the specification
- Bluetooth Qualified Test Facilities (BQTF)

11 What Does Bluetooth Wireless Technology Do?
- Cable Replacement

12 Bluetooth RF Specifications
- Specified for low cost, single chip implementation
 - Noise floor margin for substrate noise and low current LNA
 - Linearity set by near-far problem
 - In-band image allows low-cost low IF
 - VCO phase noise enables integrated VCO
 - TX-RX turn around time enables single synthesizer
 - 2.4 ISM band chosen for global use and process capabilities
- Sensitivity traded for low-cost integration of transceiver and baseband

13 Basic Baseband Protocol
- Spread spectrum frequency hopping radio
 - 79/23 one MHz channels
 - Hops every packet
 - Packets are 1, 3, or 5 slots long
 - Frame consists of two packets
 - Transmit followed by receive
 - Nominally hops at 1600 times a second (1 slot packets)

14 Network Topology
- Radio Designation

Document 1 - Page 2

- Connected radios can be master or slave
- Radios are symmetric (same radio can be master or slave)

- Piconet
 - Master can connect to 7 simultaneous or 200+ active slaves per piconet
 - Each piconet has maximum capacity (1 MSPS)
 - Unique hopping pattern/ID
- Scatternet
 - High capacity system
 - Minimal impact with up to 10 piconets within range
 - Radios can share piconets!

15 The Piconet
- All devices in a piconet hop together
 - In forming a piconet, master gives slaves its clock and device ID
 - Hopping pattern determined by device ID (48-bit)
 - Phase in hopping pattern determined by Clock
- Non-piconet devices are in standby
- Piconet Addressing
 - Active Member Address (AMA, 3-bits)
 - Parked Member Address (PMA, 8-bits)

16 Functional Overview
- Standby
 - Waiting to join a piconet
- Inquire
 - Ask about radios to connect to
- Page
 - Connect to a specific radio
- Connected
 - Actively on a piconet (master or slave)
- Park/Hold/Sniff
 - Low Power connected states

17 Packet Types/Data Rates
- ASL –Packet like behavior
- SCO – Circuit like behavior

18 Mobile = Battery Life
- Low power consumption*
 - Standby current < 0.3 mA
 - Þ 3 months

- Voice mode 8-30 mA
 - Þ 75 hours
- Data mode average 5 mA
- (0.3-30mA, 20 kbit/s, 25%)
 - Þ 120 hours
- Low Power Architecture
 - Programmable data length (else radio sleeps)
 - Hold and Park modes 60 µA
 - Devices connected but not participating
 - Hold retains AMA address, Park releases AMA, gets PMA address
 - Device can participate within 2 ms
 - *Estimates calculated with 600 mAh battery and internal amplifier, power will vary with implementation

19 Error Handling
- Forward-error correction (FEC)
 - Headers are protected with 1/3 rate FEC and HEC
 - Payloads may be FEC protected
 - 1/3 rate: simple bit repetition (SCO packets only)
 - 2/3 rate: (10,15) shortened Hamming code
 - 3/3 rate: no FEC
- ARQ (ACL packets only)
 - 16-bit CRC (CRC-CCITT) and 1-bit ACK/NACK
 - 1-bit sequence number

20 Bluetooth Security Features
- Fast Frequency Hopping (79 channels)
- Low Transmit Power (range <= 10m)
- Authentication of remote device
 - Based on link key (128 Bit)
 - May be performed in both directions
- Encryption of payload data
 - Stream cipher algorithm (≤ 128 Bit)
 - Affects all traffic on a link
- Initialization
 - PIN entry by user

21 Application Level Security
- Builds on-top of link-level security
 - Creates trusted device groups
- Security levels for services
 - Authorization required
 - Authentication required
 - Encryption required

- Different or higher security requirements could be added:
 - Personal authentication
 - Higher security level
 - Public key

22 Bluetooth Wireless Technology Is Global
- One version for the world
 - Architecture compliant with global emission rules (2.4 GHz ISM band)
 - Working through FCC, EC, MPT for spectrum, and power harmonization
 - Architecture compliant and safe for use on airlines
 - Working with FAA, JAA, FCC, airplane manufacturers, and airlines
 - Reviewing security architecture with affected countries

23 Bluetooth Radio Modules
- Complete radio on a module
 - Designed to meet "Limited Module Compliance" (LMA) requirements
 - Pre-certified to meet global regulatory requirements
 - Allows devices assembled with modules to be "self-certified"
 - USB Interface
 - Solder-ball connections
 - External Antennae

24 Bluetooth Protocols

25 Interoperability And Profiles
- Represents default solution for usage model
- Vertical slice through the protocol stack
- Basis for interoperability and logo requirements
- Each Bluetooth device supports one or more profiles

26 Summary
- Bluetooth wireless technology is a global, RF-based (ISM: 2.4GHz band), short-range, connectivity technology and solution for portable, personal devices
 - It is not just a radio
 - Create piconets on-the-fly (approximately 1Mbps)
 - Piconets may overlap in time and space for high aggregate bandwidth
- The Bluetooth spec comprises
 - A hardware and software protocol specification
 - Usage case scenario profiles and

interoperability requirements

- To learn more: http://www.bluetooth.com

27 Call To Action

- Join the SIG if you haven't already
 - Help advance Bluetooth functionality by supporting the working groups committees
 - Got a new usage model? Submit a request
- Learn how Bluetooth wireless technology works NOW!
 - See Microsoft's presentation on Bluetooth wireless technology
 - Big conference in Monte Carlo - check it out!
 - More information: http://www.Bluetooth.com
- Implement Bluetooth software and hardware in your products and systems
 - Insure interoperability via Un-plugfests
- Help support native operating system development
 - Provide test hardware to Microsoft

Confidential - Microsoft

0n2h0" 999960

APPENDIX D

Bluetooth User Experience in Windows

Dr. Michael W. Foley
Software Design Engineer
Microsoft Corporation

High Level Goals

- Easy configuration
 - Configure a new device once
 - Wizard to guide user through the process
- Ease of use
 - After initial configuration, it "just works"
 - Known devices can interoperate without user intervention
 - Don't overwhelm user with unnecessary information
- Enable advanced functionality
 - User configurable security
 - Local radio properties

User Scenarios Enabled

- Cell phone as a modem
 - Dial-up networking profile
- Digital camera to PC
 - File transfer profile
- Public Internet access points
 - LAN Access Profile
- Business card exchange
 - Object push profile
- Synchronize PPC with desktop

Configuration Issues

- Security modes
 - Open device – no security
 - Secure connections only
 - Service level security
- Binding devices
 - PIN entry on multiple devices
 - Support for devices without a UI
- Enable hidden computing
- How to hide this complexity from the average user?

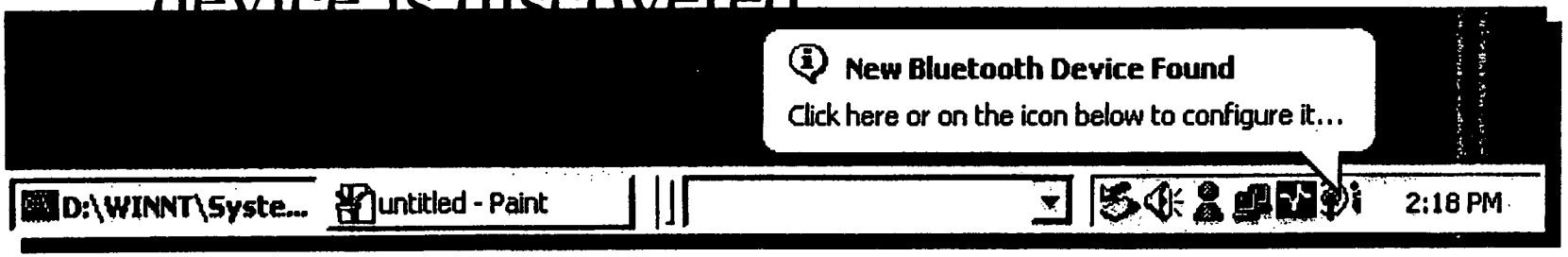
Confidential

Device States

- Unknown device in range
- Known, unbound device in range
- Bound device in range
- Device out of range

Unknown Device In Range

- Device discovery
 - Periodic polling can be enabled
 - User can force a discovery
- Unobtrusive notification will appear when a device is discovered



Managing Devices

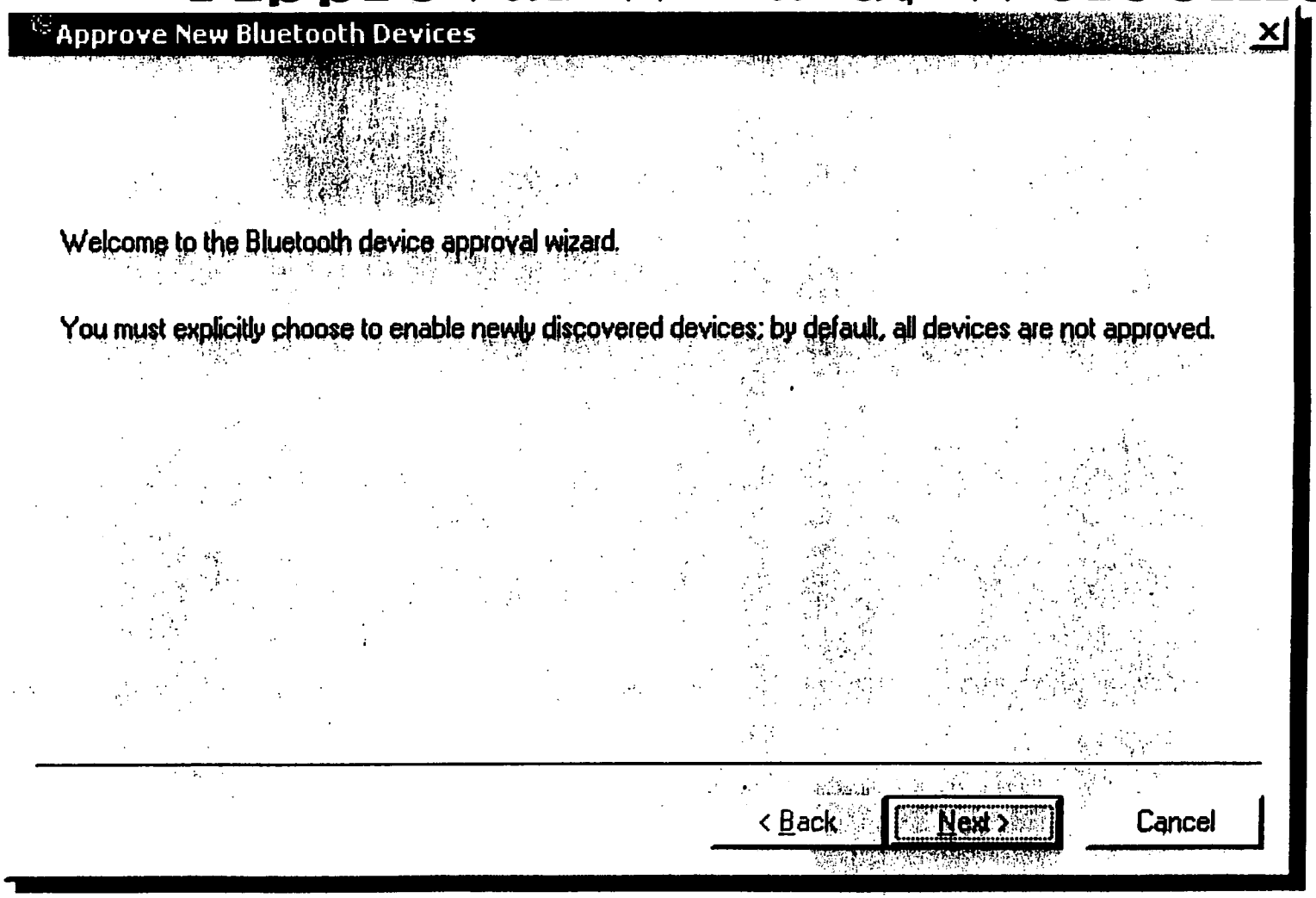
- Approve device
 - Establish trust relationship
 - Requires entering data on each device
 - One-time procedure
- Establish temporary connection
 - One-time service usage
 - May or may not be secure connection
- Instruct Windows to ignore this device in the future
 - Stop telling me about cell phone in the office next to mine

0012h0" 9999960

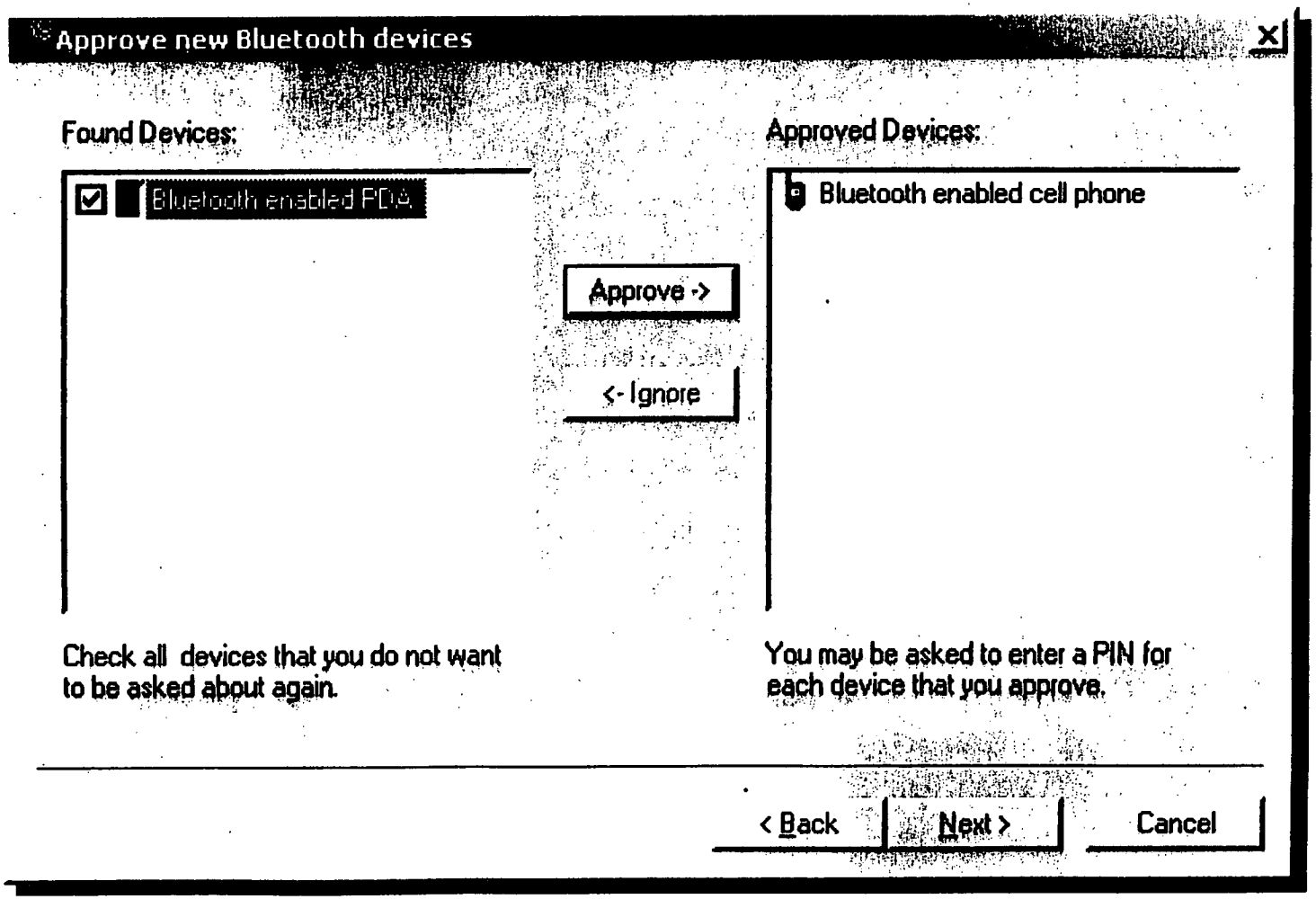
Approval Wizard

- Guides the user through the binding process
- Welcome page
- Brief explanation of what the user needs to do
- Specifically calls out that devices must be explicitly enabled

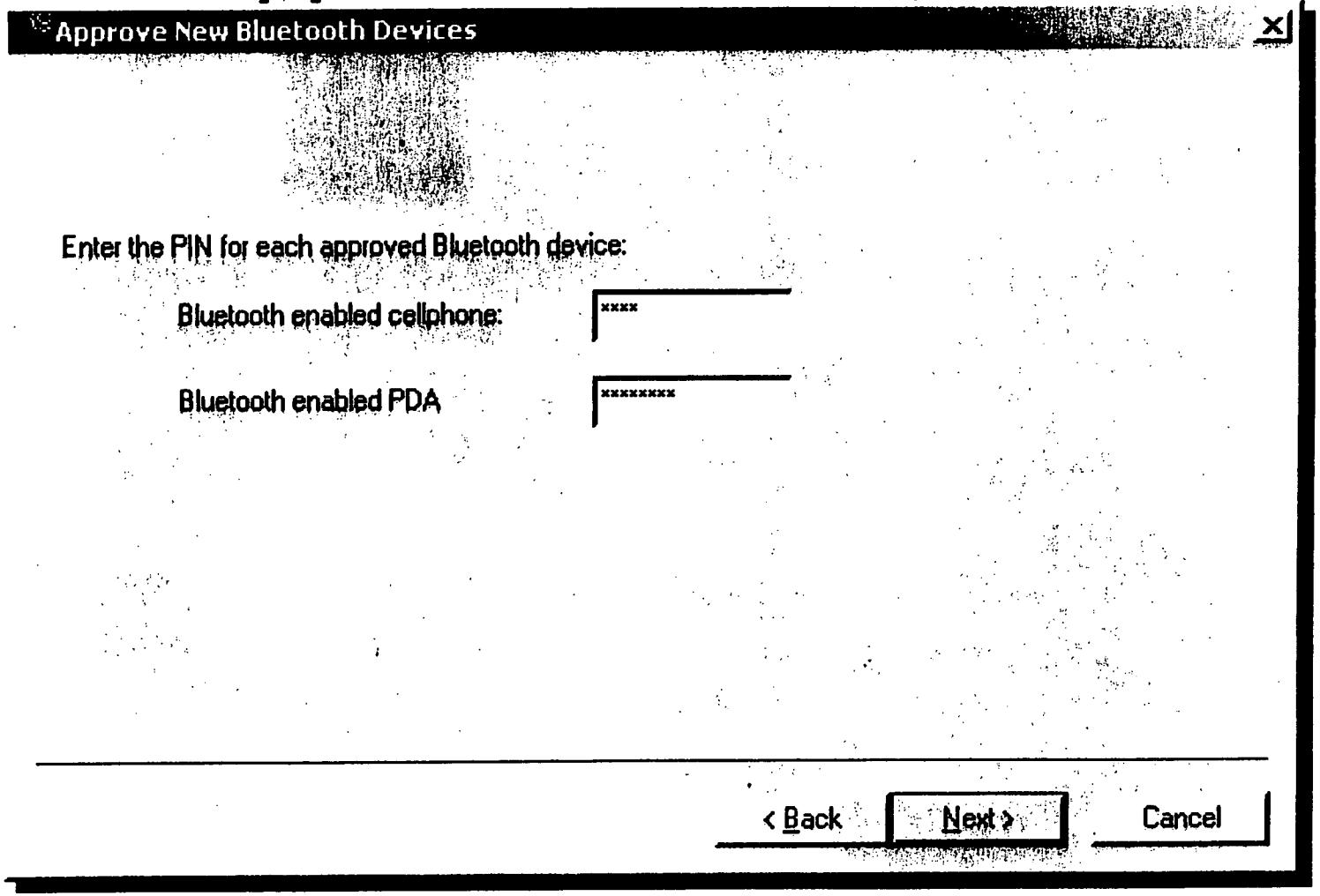
Approval Wizard, Welcome



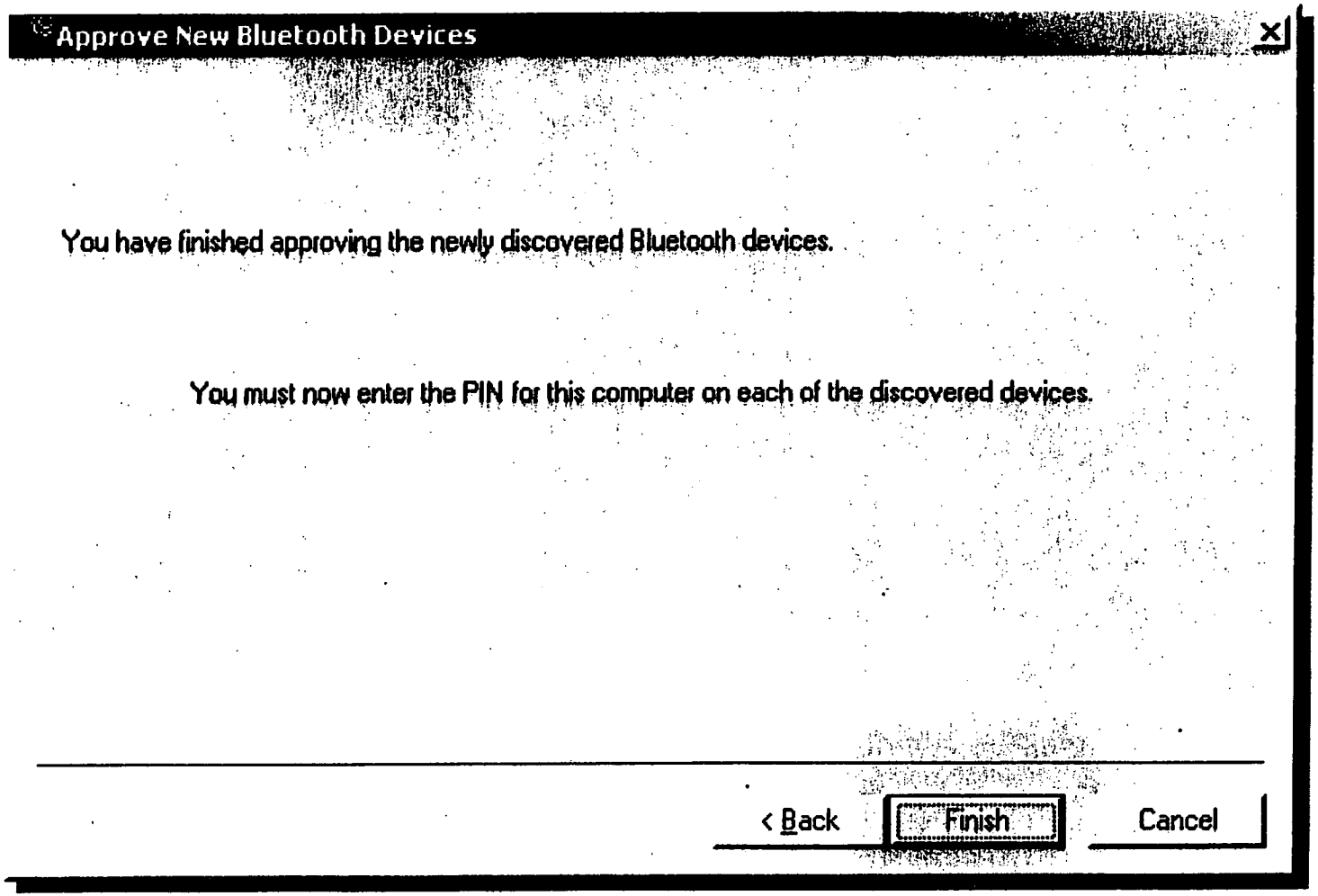
Approval Wizard, Discovered Devices



Approval Wizard, Key Entry



Approval Wizard, Remote Device Instructions

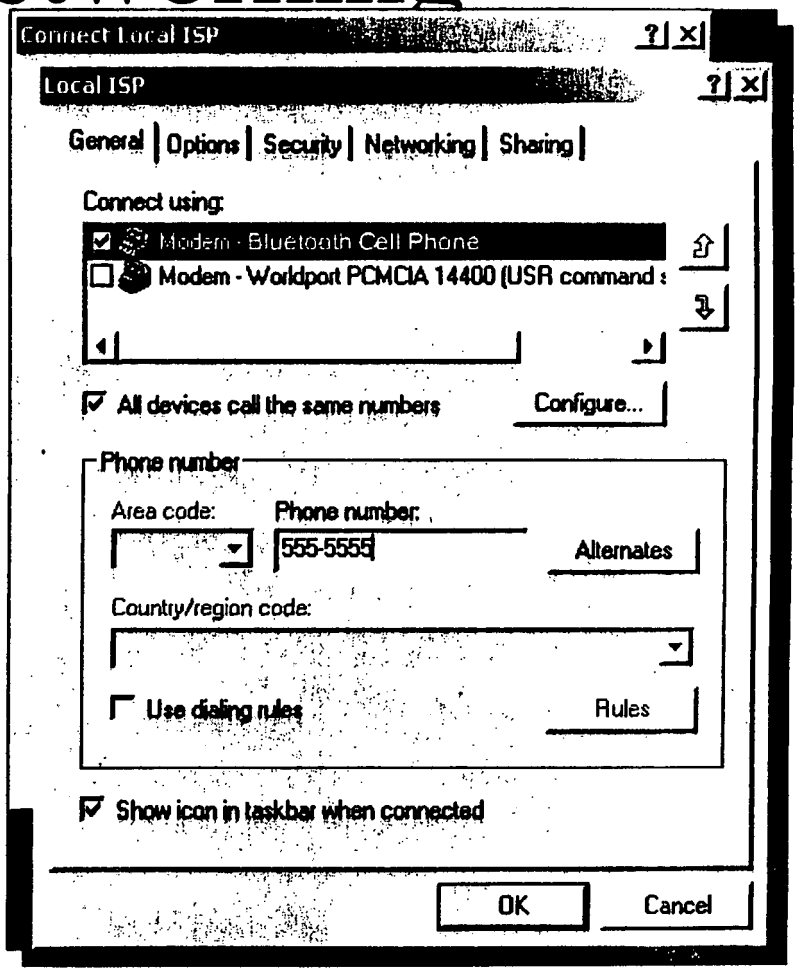


Approved Device In Range

- User can immediately use services exported by the device
 - PIN not required
- Supports "Hidden Computing" model
 - PPC automatically syncs with desktop when in range

Dialup Networking

- Use the Network Connection Wizard to create a connectiod
- Select the Bluetooth Device that appears as a modem
- If a connectiod already exist, simply change to the Bluetooth modem device
- Connect to network just like any other modem

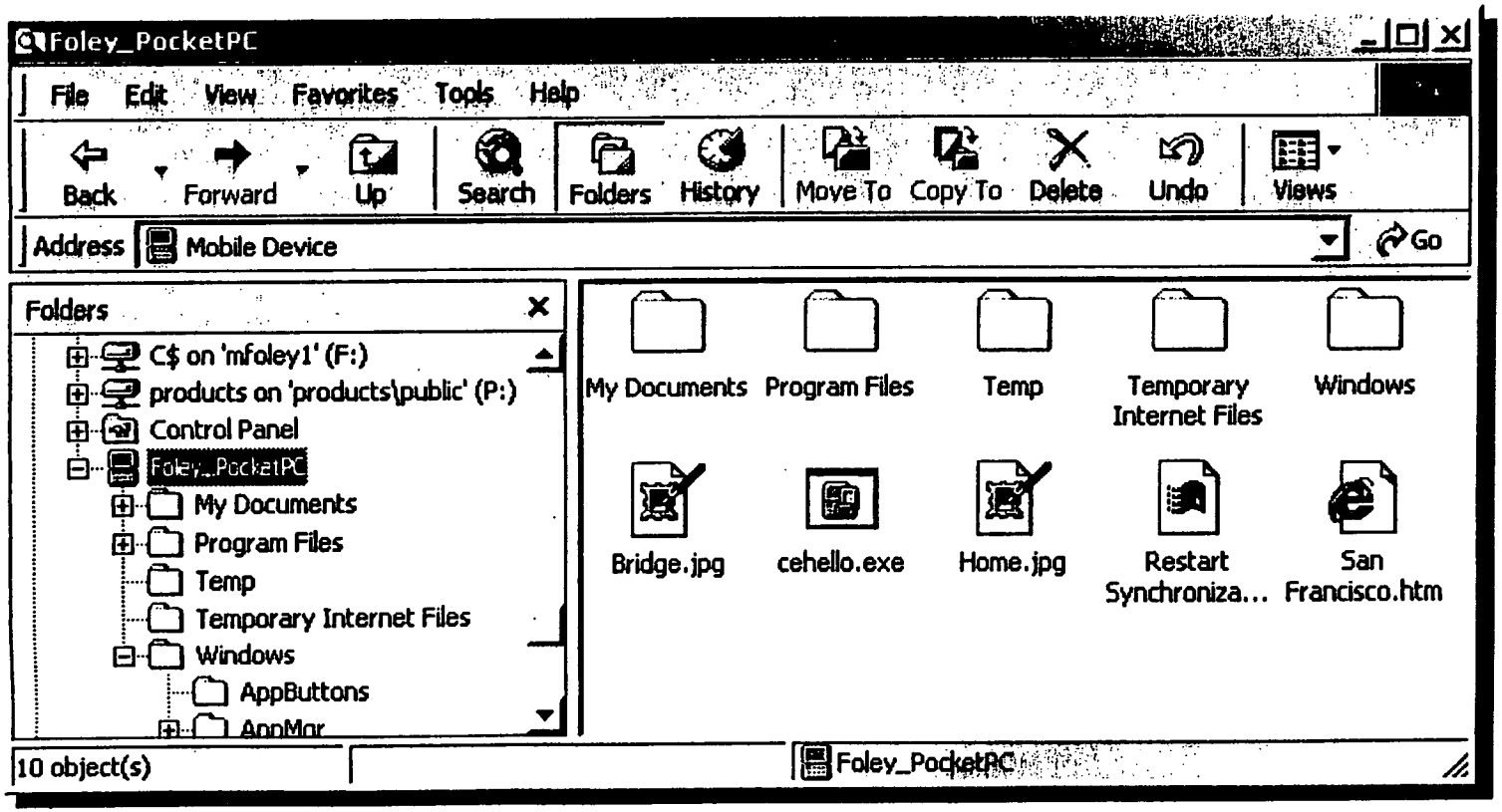


LAN Access Point

- LAN Access Point looks like a modem connection
- Usage is the same as DUN, except the connectoid connects to the LAN access point rather than the phone/modem

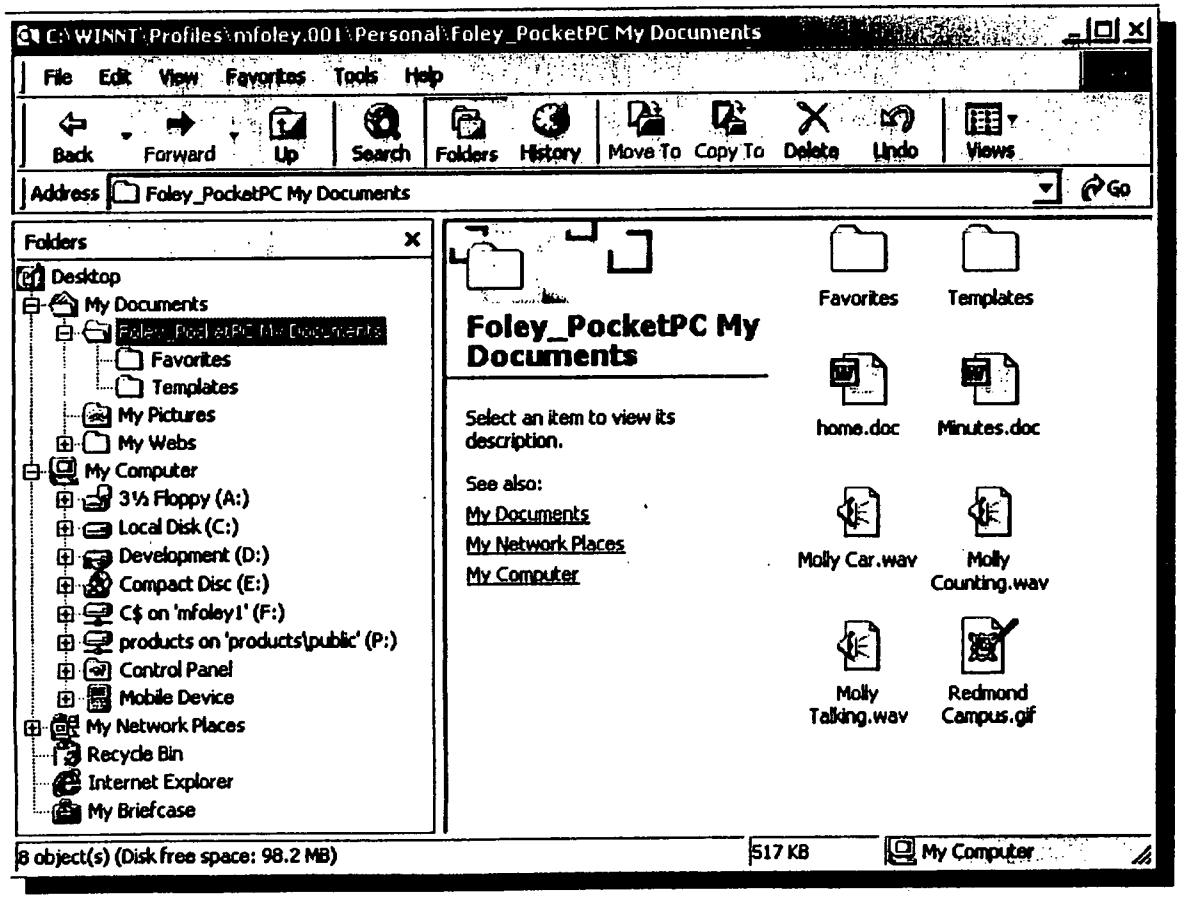
File Transfer

- Browser remote device



Synchronization

- File as well as contacts, appointments, task, etc.



0402HO 999960

Advanced Functionality

- Listing all devices in range
- Managing local services
- Managing local radio parameters
- Removing devices
- Forced discovery

Remote Device Browsing

- User can view all approved devices; both in and out of range
 - Can remove device from approved status
- User can only view unapproved devices currently in range
 - History of all devices seen isn't stored
- Explore Services on remote devices
- Property page for each device will display URLs associated with service discovery record for the device

0n2n0" 5959560

Managing Local Services

- View currently exported services
- Alter security settings
 - Must be bound device by default
 - Enable access to unbound devices
 - Disable service to all devices

Managing Local Radio Properties

- Set power management policies
 - AC Powered versus Battery Powered
 - Periodic polling frequency
 - Machine is on
 - Machine is in standby
 - Machine is in hibernate
 - Which COD wake machine
- Set filtering rules
 - Discover particular CODs
 - Discover particular radios
- Some properties are read-only for non administrators

Oth2hO' 9999960

Summary

- Designed for ease of use
 - Configure devices once
 - More information and control available when required
- Integrated Windows User Interface
- Many usage models supported out of the box
 - Dial-up networking
 - Synchronization
 - File Transfer
 - LAN Access

Call To Action

- Beta test our software
 - Learn first hand what is intuitive, and what is not
 - Which features should be more readily available
 - Which features are rarely used
- Provide us feedback
- Provide us hardware
 - Allows us to test the user experience with a wide range of scenarios

1 Bluetooth User Experience in Windows

Dr. Michael W. Foley
Software Design Engineer
Microsoft Corporation

2 High Level Goals

- Easy configuration
 - Configure a new device once
 - Wizard to guide user through the process
- Ease of use
 - After initial configuration, it "just works"
 - Known devices can interoperate without user intervention
 - Don't overwhelm user with unnecessary information
- Enable advanced functionality
 - User configurable security
 - Local radio properties

3 User Scenarios Enabled

- Cell phone as a modem
 - Dial-up networking profile
- Digital camera to PC
 - File transfer profile
- Public Internet access points
 - LAN Access Profile
- Business card exchange
 - Object push profile
- Synchronize PPC with desktop

4 Configuration Issues

- Security modes
 - Open device – no security
 - Secure connections only
 - Service level security
- Binding devices
 - PIN entry on multiple devices
 - Support for devices without a UI
- Enable hidden computing
- How to hide this complexity from the average user?

5 Device States

- Unknown device in range
- Known, unbound device in range
- Bound device in range
- Device out of range

6 Unknown Device In Range
- Device discovery
 - Periodic polling can be enabled
 - User can force a discovery
- Unobtrusive notification will appear when a device is discovered

7 Managing Devices
- Approve device
 - Establish trust relationship
 - Requires entering data on each device
 - One-time procedure
- Establish temporary connection
 - One-time service usage
 - May or may not be secure connection
- Instruct Windows to ignore this device in the future
 - Stop telling me about cell phone in the office next to mine

8 Approval Wizard
- Guides the user through the binding process
- Welcome page
- Brief explanation of what the user needs to do
- Specifically calls out that devices must be explicitly enabled

9 Approval Wizard, Welcome

10 Approval Wizard, Discovered Devices

11 Approval Wizard, Key Entry

12 Approval Wizard, Remote Device Instructions

13 Approved Device In Range
- User can immediately use services exported by the device
 - PIN not required
- Supports "Hidden Computing" model
 - PPC automatically syncs with desktop when in range

14 Dialup Networking

15 LAN Access Point
- LAN Access Point looks like a modem connection
- Usage is the same as DUN, except the connectoid connects to the LAN access point rather than the phone/modem

16 File Transfer
- Browser remote device
- Drag and drop files

17 Synchronization
- File as well as contacts, appointments, task, etc.

18 Advanced Functionality
- Listing all devices in range

- Managing local services
- Managing local radio parameters
- Removing devices
- Forced discovery

19 Remote Device Browsing
- User can view all approved devices; both in and out of range
 - Can remove device from approved status
- User can only view unapproved devices currently in range
 - History of all devices seen isn't stored
- Explore Services on remote devices
- Property page for each device will display URLs associated with service discovery record for the device

20 Managing Local Services
- View currently exported services
- Alter security settings
 - Must be bound device by default
 - Enable access to unbound devices
 - Disable service to all devices

21 Managing Local Radio Properties
- Set power management policies
 - AC Powered versus Battery Powered
 - Periodic polling frequency
 - Machine is on
 - Machine is in standby
 - Machine is in hibernate
 - Which COD wake machine
- Set filtering rules
 - Discover particular CODs
 - Discover particular radios
- Some properties are read-only for non administrators

22 Summary
- Designed for ease of use
 - Configure devices once
 - More information and control available when required
- Integrated Windows User Interface
- Many usage models supported out of the box
 - Dial-up networking
 - Synchronization
 - File Transfer
 - LAN Access

23 Call To Action

- Beta test our software
 - Learn first hand what is intuitive, and what is not
 - Which features should be more readily available
 - Which features are rarely used
- Provide us feedback
- Provide us hardware
 - Allows us to test the user experience with a wide range of scenarios

0555555 - 0000003

APPENDIX E

Bluetooth Stack In Windows

Ken Ray and Stan Adermann
Software Development Engineers
Windows Division
Microsoft Corporation

Agenda

- Bluetooth Architecture in Windows
 - Goals
 - Components of the Stack
 - Functionality
- Opportunities for IHVs and ISVs
 - Applications
 - Services
 - Devices

0n2h0" 5959560

High Level Goals

- PC work with all devices
 - Bluetooth Devices as PC peripherals
 - Bluetooth Devices as PC companions
 - Bluetooth Devices bridge to network resources through a PC
- Easy to configure and operate
- Extensible architecture
 - Platform for third parties to add value

0n2n0' 5959560

Scenarios

- Device configuration:
 - Discovery
 - Bonding
- Syncing and transfer through OBEX
 - Files
 - Pictures
 - Vcards
- Dial up Networking
 - Cell as modem
 - Null Modem for Peer to peer
- Generic RFComm applications
 - Non-OBEX synchronization
 - Other serial-type applications

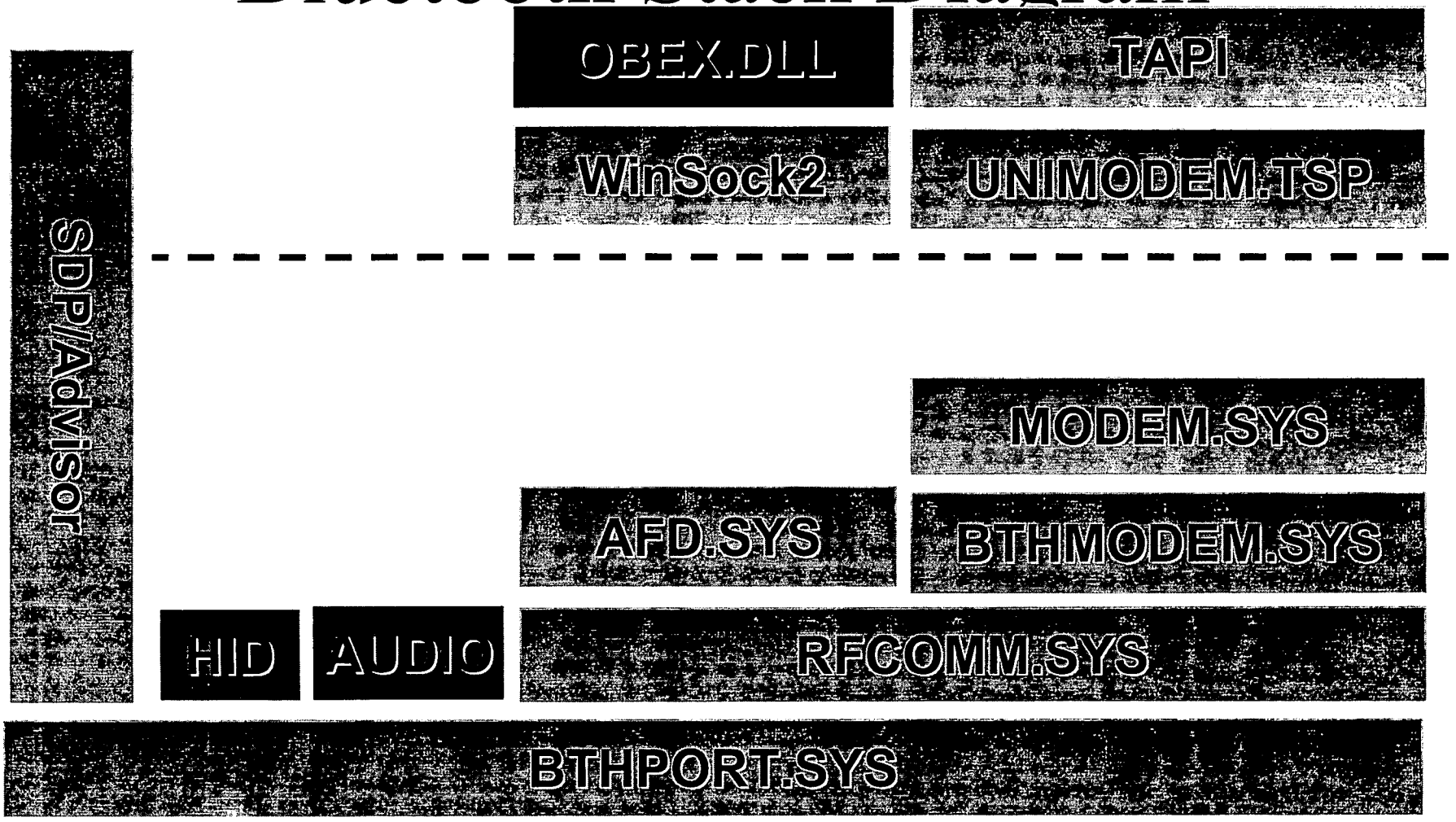
0013h0" 5555550

Technical Requirements

- Bluetooth 1.0 Type II device classification supported
- Required profiles
- Bus Management Infrastructure
 - Device and radio configuration
 - Control panels
 - System Trays
- Extensible framework for value adds
 - Devices
 - Profiles
 - Bus mgmt software
 - RFComm applications
 - Object Exchange and special object handling
 - RAS and TAPI over Unimodem

0n8n0" 5555560

Bluetooth Stack Diagram



Stack Components

- BthPort
 - L2Cap / HCI
 - Hardware abstraction: Serial, USB…
 - Enumeration of Found Bound Services
- SDP/Management UI
 - Bus management:
 - User notification of newly discovered devices
 - User assisted Configuration and Bonding
 - Configuration of radio
 - Local Service Exposure and Publication

Stack Components

- RFCOMM
 - RFComm Profile
 - TDI interface for WinSock (AFD)
 - Bus enumeration for DUNs
- BthModem (a WDM modem)
- OBEX.DLL
 - Object Exchange 1.2
 - Bus Agnostic

0n2n0" 9999560

BthPort

- Support Currently Defined buses: USB, Serial, 16550
- Plug and Play events
- Bluetooth Request Blocks

SDP

- Provide a "builder" interface to easily create a service record
- Kernel mode
 - Client drivers can submit a list of UUIDs to search for on all newly discovered devices or initiate a SDP search outside of device discovery
 - BThPort will search for all the services in the browse group hierarchy
- User mode
 - Initiate searches
 - Browse service records

Management UI

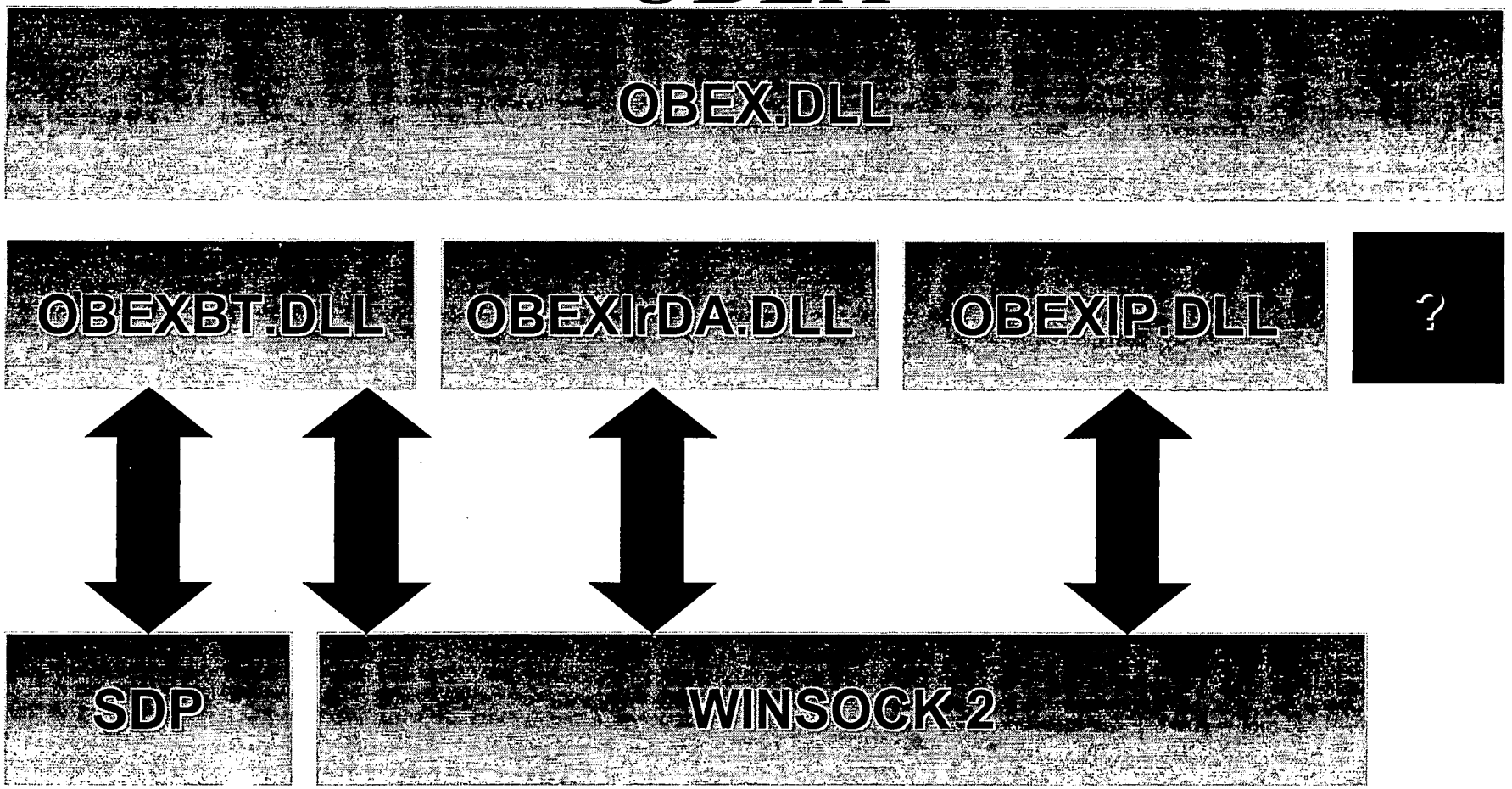
- Present user with devices in range and bound devices
- Allows the user to easily change the relationship with remote device
- Provide unobtrusive PIN and authorization notifications
- UI is accessible from third-party applications for a standard user experience
- Advanced features
 - Filter devices based on COD or address
 - Local radio settings
 - Manage power policies

Confidential

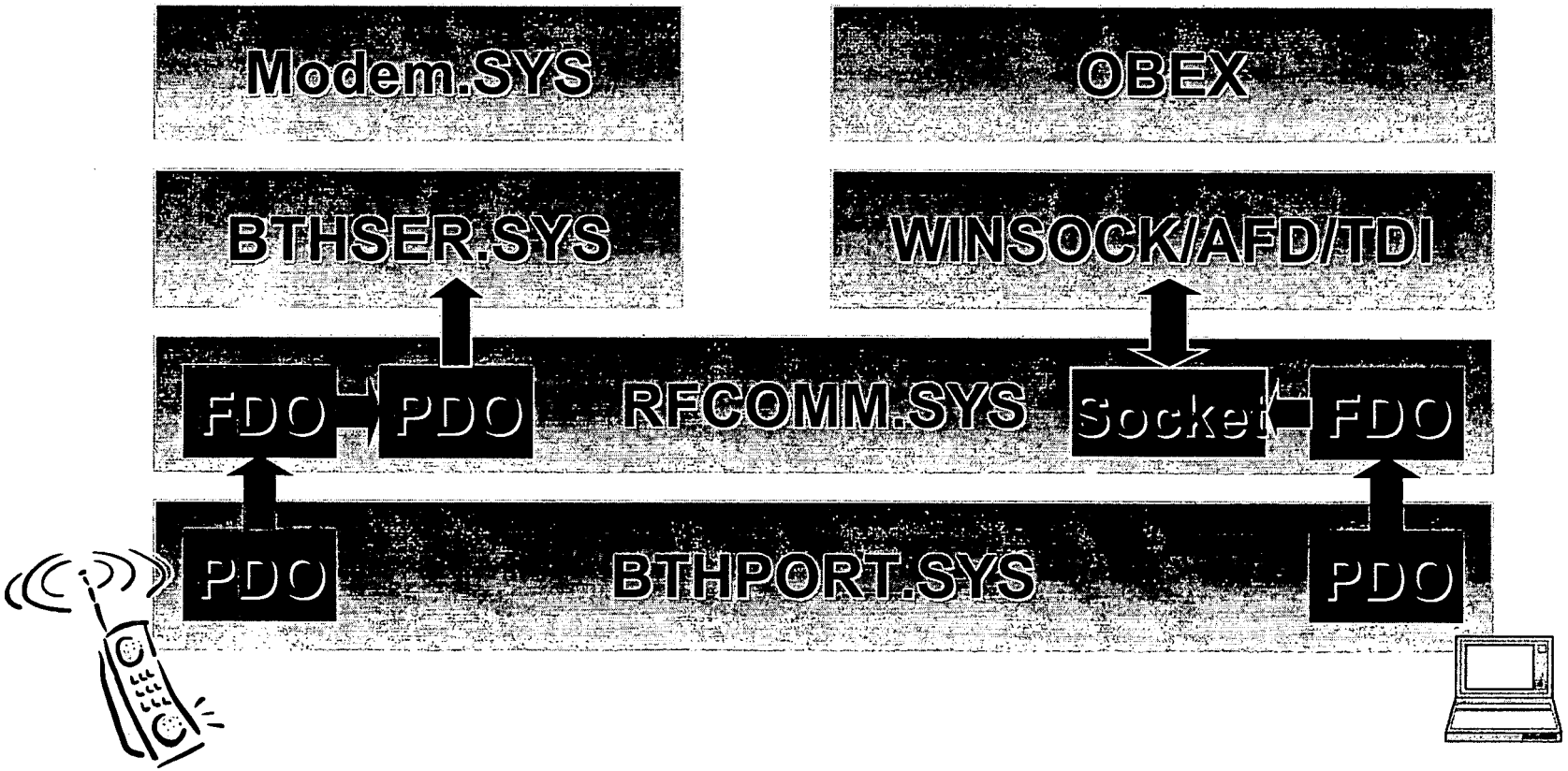
OBEX

- Full OBEX 1.2 implementation:
 - Put
 - Get
 - SetPath
 - Definable transactions
- COM API
- Extensible to other media and transports

OBEX



RFCOMM



DUN/Lan Access

File Transfer/OBEX

Opportunities To Add Value

- RF comm applications
- OBEX applications/extensions
- Bluetooth management application
- New device types and/or class drivers
- Radios on new hardware buses

0012ad 9999560

RF Comm Applications

- Applications looking for virtual serial ports not supported
- Legacy TAPI/Unimodem applications see peer devices as NULL Modems
- Applications enumerate Modem/Serial Devices through Unimodem

Confidential

RF Comm Applications

- Winsock allows for dynamic discovery and communication
 - Talk to the device, not to the conduit ("My Laserjet" versus LPT2)
 - Once bonded device is in range the application can find and use it
 - Allows for multiple remote connection to same service
 - Not necessary to manage multiple virtual COMx ports

OBEX Applications

- Examples
 - Photos
 - Vcards (not “in the box”)
 - Simple databases
- Server
 - Registration
 - New Obex Commands and types
 - Application can register as handler for custom commands
- Client
 - Discovery
 - Navigate directory structure (enumerate objects)
 - Push Pull objects

Confidential

Bluetooth Management Applications

- Substitution of stock Microsoft Plug and Play experience
 - Configuration and bonding of devices

New Profiles Types

- Native L2CAP
- Examples:
 - HID
 - Remote NDIS
 - Doom Server with Streaming Audio (utilize native audio channels)

New Profile Types

- Server
 - Registers with SDP/Adviser module
 - No Plug and Play event until remote peer connects
 - PDO means active connection on which local server driver loaded
- Client
 - BthPort finds remote service
 - Signals SDP/Advisor to determine if user wants to use this device (Approval Wizard)
 - Plug and Play event (PDO) means active connection on which local client driver loaded

Confidential

New Hardware Buses

- Examples
 - Register set for Card Bus/PCI
 - 1394
- Miniport

8n2n0' 9999960

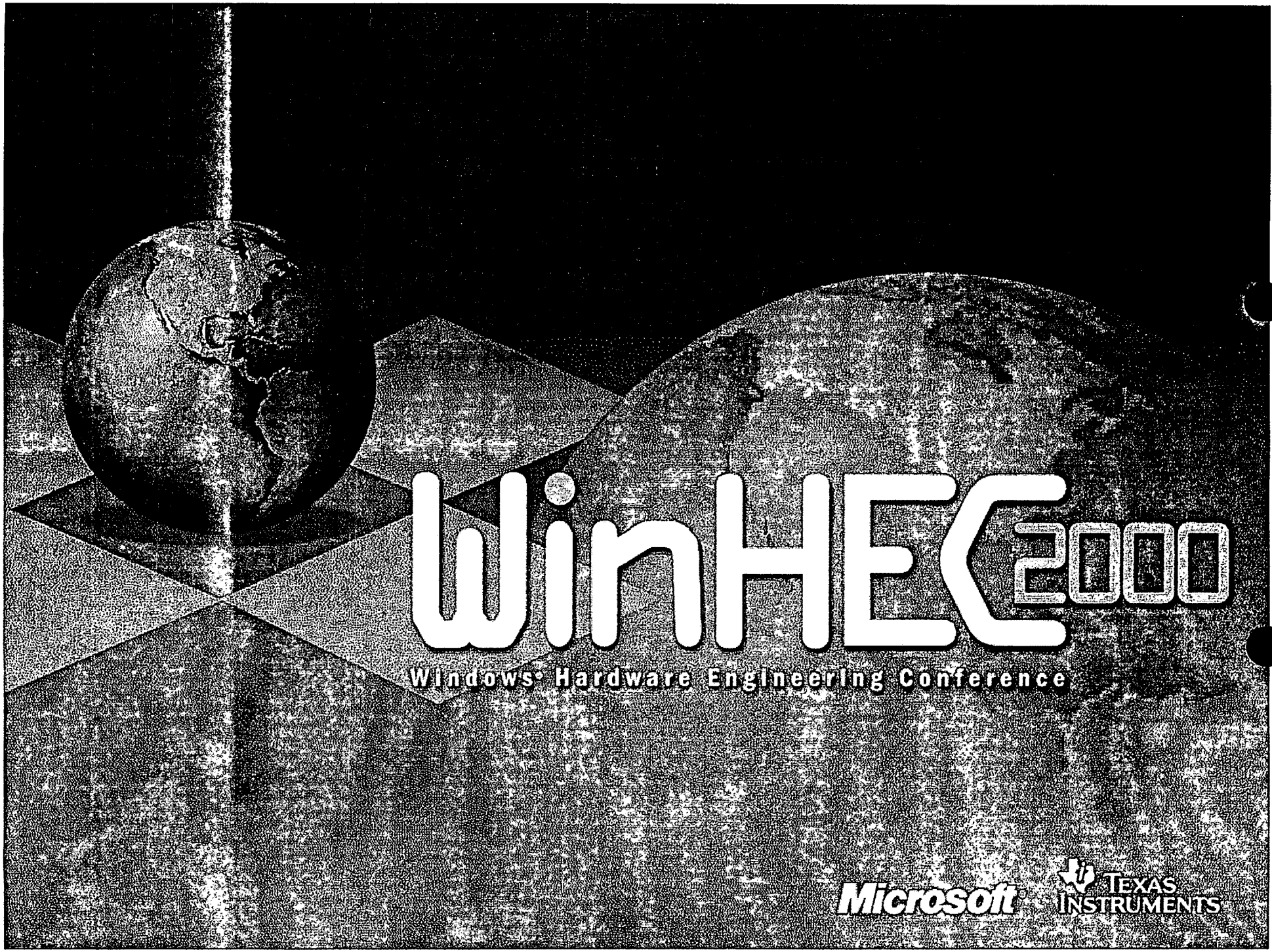
Calls To Action

- Is your value add method missing?
- We need your hardware
 - Phones
 - Radios
 - Phones
 - Peripherals
 - Phones
 - PDAs
- We need your software
 - Applications and drivers
 - Can we upgrade you?
- Come to the developers' conference

Confidential

References

- http://www.microsoft.com/hwdev/bluetooth
- Contact: BTINFO@Microsoft.COM


WinHEC2000
Windows® Hardware Engineering Conference
Microsoft
Texas Instruments

1

2 Agenda

- Bluetooth Architecture in Windows
 - Goals
 - Components of the Stack
 - Functionality
- Opportunities for IHVs and ISVs
 - Applications
 - Services
 - Devices

3 High Level Goals

- PC work with all devices
 - Bluetooth Devices as PC peripherals
 - Bluetooth Devices as PC companions
 - Bluetooth Devices bridge to network resources through a PC
- Easy to configure and operate
- Extensible architecture
 - Platform for third parties to add value

4 Scenarios

- Device configuration:
 - Discovery
 - Bonding
- Syncing and transfer through OBEX
 - Files
 - Pictures
 - Vcards
- Dial up Networking
 - Cell as modem
 - Null Modem for Peer to peer
- Generic RFComm applications
 - Non-OBEX synchronization
 - Other serial-type applications

5 Technical Requirements

- Bluetooth 1.0 Type II device classification supported
- Required profiles
- Bus Management Infrastructure
 - Device and radio configuration
 - Control panels
 - System Trays
- Extensible framework for value adds
 - Devices

- Profiles
- Bus mgmt software
- RFComm applications
- Object Exchange and special object handling
- RAS and TAPI over Unimodem

6 Bluetooth Stack Diagram

7 Stack Components
- BthPort
 - L2Cap / HCI
 - Hardware abstraction: Serial, USB...
 - Enumeration of Found Bound Services
- SDP/Management UI
 - Bus management:
 - User notification of newly discovered devices
 - User assisted Configuration and Bonding
 - Configuration of radio
 - Local Service Exposure and Publication

8 Stack Components
- RFCOMM
 - RFComm Profile
 - TDI interface for WinSock (AFD)
 - Bus enumeration for DUNs
- BthModem (a WDM modem)
- OBEX.DLL
 - Object Exchange 1.2
 - Bus Agnostic

9 BthPort
- Support Currently Defined buses: USB, Serial, 16550
- Plug and Play events
- Bluetooth Request Blocks

10 SDP
- Provide a "builder" interface to easily create a service record
- Kernel mode
 - Client drivers can submit a list of UUIDs to search for on all newly discovered devices or initiate a SDP search outside of device discovery
 - BThPort will search for all the services in the browse group hierarchy
- User mode
 - Initiate searches
 - Browse service records

11 Management UI
- Present user with devices in range and bound devices

- Allows the user to easily change the relationship with remote device
- Provide unobtrusive PIN and authorization notifications
- UI is accessible from third-party applications for a standard user experience
- Advanced features
 - Filter devices based on COD or address
 - Local radio settings
 - Manage power policies

12 OBEX
- Full OBEX 1.2 implementation:
 - Put
 - Get
 - SetPath
 - Definable transactions
- COM API
- Extensible to other media and transports

13 OBEX

14 RFCOMM

15 Opportunities To Add Value
- RF comm applications
- OBEX applications/extensions
- Bluetooth management application
- New device types and/or class drivers
- Radios on new hardware buses

16 RF Comm Applications
- Applications looking for virtual serial ports not supported
- Legacy TAPI/Unimodem applications see peer devices as NULL Modems
- Applications enumerate Modem/Serial Devices through Unimodem

17 RF Comm Applications
- Winsock allows for dynamic discovery and communication
 - Talk to the device, not to the conduit ("My Laserjet" versus LPT2)
 - Once bonded device is in range the application can find and use it
 - Allows for multiple remote connection to same service
 - Not necessary to manage multiple virtual COMx ports

18 OBEX Applications
- Examples
 - Photos
 - Vcards (not "in the box")
 - Simple databases
- Server
 - Registration

- New Obex Commands and types
- Application can register as handler for custom commands

- Client
 - Discovery
 - Navigate directory structure (enumerate objects)
 - Push Pull objects

19 ☐ Bluetooth Management Applications

- Substitution of stock Microsoft Plug and Play experience
 - Configuration and bonding of devices

20 ☐ New Profiles Types

- Native L2CAP
- Examples:
 - HID
 - Remote NDIS
 - Doom Server with Streaming Audio (utilize native audio channels)

21 ☐ New Profile Types

- Server
 - Registers with SDP/Adviser module
 - No Plug and Play event until remote peer connects
 - PDO means active connection on which local server driver loaded
- Client
 - BthPort finds remote service
 - Signals SDP/Advisor to determine if user wants to use this device (Approval Wizard)
 - Plug and Play event (PDO) means active connection on which local client driver loaded

22 ☐ New Hardware Buses

- Examples
 - Register set for Card Bus/PCI
 - 1394
- Miniport

23 ☐ Calls To Action

- Is your value add method missing?
- We need your hardware
 - Phones
 - Radios
 - Phones
 - Peripherals
 - Phones
 - PDAs

Microsoft Confidential

- We need your software
 - Applications and drivers
 - Can we upgrade you?
- Come to the developers' conference

24 References

- http://www.microsoft.com/hwdev/bluetooth
- Contact: BTINFO@Microsoft.COM

25

0356565 .0420

APPENDIX F

Remote NDIS Over Bluetooth Specification

Rev 0.15
March 20, 2000

© 2000 Microsoft Corporation. All rights reserved.

Microsoft Confidential

1 INTRODUCTION 3
1.1 REVISION HISTORY 3
1.2 OPEN ISSUES 3
1.3 LICENSE AGREEMENT 4
2 REMOTE NDIS TO BLUETOOTH MAPPING 5
2.1 DATA ENCAPSULATION 6
2.1.1 Bluetooth L2CAP Channels *6*
2.1.2 Bluetooth Packets *6*
2.1.3 Bluetooth Flow Control *6*
2.2 REMOTE NDIS DEVICE SUPPORT 6
2.2.1 Service Discovery Descriptor *6*
2.2.2 L2Cap Channels *7*
2.2.3 Messages *7*
2.3 BLUETOOTH PEER-TO-PEER RNDIS 7
2.3.1 REMOTE_NDIS_INITIALIZE_MSG *8*
2.3.2 REMOTE_NDIS_QUERY_MSG *8*
2.3.2.1 OID_GEN_SUPPORTED_LIST 8
2.3.2.2 OID_GEN_HARDWARE_STATUS 8
2.3.2.3 OID_GEN_MEDIA_SUPPORTED 8
2.3.2.4 OID_GEN_MEDIA_IN_USE 8
2.3.2.5 OID_GEN_MAXIMUM_FRAME_SIZE 8
2.3.2.6 OID_GEN_LINK_SPEED 8
2.3.2.7 OID_GEN_TRANSMIT_BLOCK_SIZE 8
2.3.2.8 OID_GEN_RECEIVE_BLOCK_SIZE 9
2.3.2.9 OID_GEN_VENDOR_ID 9
2.3.2.10 OID_GEN_VENDOR_DESCRIPTION 9
2.3.2.11 OID_GEN_CURRENT_PACKET_FILTER 9
2.3.2.12 OID_GEN_MAXIMUM_TOTAL_SIZE 9
2.3.2.13 OID_GEN_MAC_OPTIONS 9
2.3.2.14 OID_GEN_MEDIA_CONNECT_STATUS 9
2.3.2.15 OID_GEN_XMIT_OK 9
2.3.2.16 OID_GEN_RCV_OK 9
2.3.2.17 OID_GEN_XMIT_ERROR 9
2.3.2.18 OID_GEN_RCV_ERROR 9
2.3.2.19 OID_GEN_RCV_NO_BUFFER 9
2.3.2.20 OID_802_3_PERMANENT_ADDRESS 9
2.3.2.21 OID_802_3_CURRENT_ADDRESS 10
2.3.2.22 OID_802_3_MULTICAST_LIST 10
2.3.2.23 OID_802_3_MAXIMUM_LIST_SIZE 10
2.3.2.24 OID_802_3_RCV_ERROR_ALIGNMENT 10
2.3.2.25 OID_802_3_XMIT_ONE_COLLISIONS 10
2.3.2.26 OID_802_3_MORE_COLLISIONS 10
2.3.3 REMOTE_NDIS_SET_MSG *10*
2.3.3.1 OID_802_3_MULTICAST_LIST 10
2.3.3.2 OID_GEN_CURRENT_PACKET_FILTER 10
2.3.4 REMOTE_NDIS_RESET_MSG *10*
2.3.5 REMOTE_NDIS_KEEPALIVE_MSG *10*

1 Introduction

This document defines how "Remote NDIS," bus-independent message set for bus-attached networking devices (either Ethernet 802.3 devices or connection oriented devices where the call manager (signaling stack) resides either on the host or on the device) is carried over Bluetooth.

1.1 Revision History

Revision	Date	Comment
0.1	2000/03/16	First draft
0.11	2000/03/17	Add microport spec to allow peer to peer working
0.12	2000/03/17	Allow either host or device to initiate L2CAP channels Add more explanation as to why peer-to-peer microport A better explanation on how to handle the reset message Tidy up some of the OID handling text to make it generic for both peer and device modes Update the main text to allow peer-to-peer operation as the Bluetooth PAN profile Add explanation of why a separate L2CAP control channel Add additional requirements for the microport to support all of the OIDs locally so allowing the remote device to support no OIDS.
0.13	2000/03/18	Updated OIDs text to allow all OIDs to be optional, but the microport needs to support them locally.
0.14	2000/03/19	Changed MAXIMUM_FRAME_SIZE OID to use L2CAP MTU rather than maximum Ethernet frame size.
0.15	2000/03/20	Talk about RNDIS_PACKET, what additional overhead it adds above raw Ethernet frames and how to optimize the overhead. Modified response to a REMOTE_NDIS_RESET_MSG

1.2 Open Issues

1.3 License Agreement

The Remote NDIS Specification and any accompanying materials (the "Specification") provided by Microsoft is for your personal use only, and may be used solely for the purpose of implementing the Remote NDIS protocol message set to interface with (i) a Microsoft Windows operating system or (ii) a bus or network-connected communications device, such as a USB, 1394, Bluetooth or TCP/IP device. **THE SPECIFICATION MAY NOT BE COPIED OR DISTRIBUTED.**

Microsoft may have copyrights, patents or pending patent applications covering subject matter in the Specification. To the extent Microsoft has such copyrights, patents or applications, Microsoft agrees to grant a nonexclusive, royalty-free, world-wide license under these copyrights, patents or applications solely to implement the Remote NDIS Specification to interface with (i) a Microsoft Windows operating system or (ii) a bus or network-connected communications device, such as a USB, 1394, Bluetooth or TCP/IP device, on condition that you agree not to assert any intellectual property rights against Microsoft or other companies for their implementation of the Specification. Microsoft reserves all other rights it may have in the Specification. The furnishing of this document does not give you any license to any other Microsoft patents, trademarks, copyrights, or other intellectual property rights. Specifically, neither this document nor the Specification give you any license to the NDIS Specification or to USB Communication Device Class technology.

The Specification is provided "AS IS" without warranty of any kind. To the maximum extent permitted by applicable law, Microsoft further disclaims all warranties, including without limitation any implied warranties of merchantability and fitness for a particular purpose, as well as warranties of title and noninfringement. The entire risk arising out of the use or performance of the Specification remains with you.

To the maximum extent permitted by applicable law, in no event shall Microsoft or its suppliers be liable for any consequential, incidental, direct, indirect, special, punitive, or other damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use the Specification, even if Microsoft has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

2 Remote NDIS to Bluetooth Mapping

Using Bluetooth L2Cap Channels, Remote NDIS manages a networking device at a higher level of abstraction than previously defined models, based on the messaging protocol used in Microsoft NDIS networking API. Remote NDIS passes encapsulated Remote NDIS control messages between the host and device over a L2CAP channel. In addition, this document explains how to use Remote NDIS control messages in a peer-to-peer mode of operation to allow emulated Ethernet point-to-point connections between two Bluetooth devices.

NOTE: An understanding of the Bluetooth 1.0 document is highly recommended before proceeding. The 1.0 document can be found at *http://www.bluetooth.com*.

Remote NDIS will leverage the current NDIS network stack and provide easy migration of networking devices to external buses for Windows machines. The Remote NDIS model has many advantages: extensibility without change to the bus specific message transport mechanisms, the ability to support more protocols over more buses quickly, and a driver architecture that has been proven for both networking and external bus device models. Microsoft will supply all driver layers for both Windows 2000 and Windows 98. The value-add mechanisms that already exist in the NDIS network stack will be supported for Remote NDIS devices.

A single L2CAP channel is used for Remote NDIS control messages, their responses and for the device to indicate a change of status. A separate L2CAP channel is used for control, to limit the latency in sending control messages. A data message can be up to 1500 bytes in length, which takes approximately 20ms when using the full Bluetooth bandwidth and will increase as other traffic uses part of the bandwidth.

Another L2CAP channel is used by Remote NDIS to exchange Remote NDIS data packets. Additional L2CAP channels may be used to deal with multiple networking channels that may exist on a device.

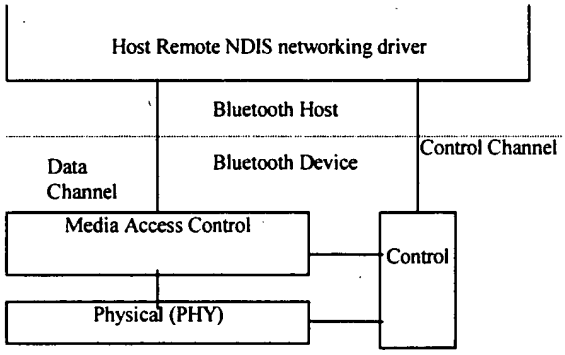


Figure 1: Remote NDIS Model

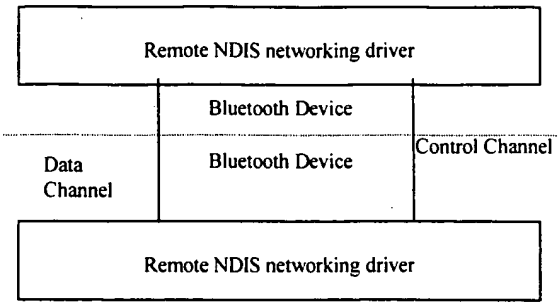


Figure 2: Remote NDIS peer-to-peer Model

A Remote NDIS device may be "connectionless" or "connection oriented" and may require connection management requests as well as device management requests to control the operational state and configuration of the device.

It is out of the scope of this document how IP addresses are assigned to this interface. This document also does not discuss how multiple links could be joined together to form a larger network by adding a L2 bridge or a L3 router above the links to allow adhoc networks to form.

2.1 Data Encapsulation

Network data is passed between the host and device over Bluetooth L2CAP channels. This data is not passed in a "raw" form, but instead encapsulated in an NDIS packet mechanism described in Section 7.1, *NDISPacket of the Remote NDIS Specification*. This format is based on the model already used in the NDIS network stack. Mechanisms for passing out-of-band data to the device are also provided for with the Remote NDIS packet definition.

2.1.1 Bluetooth L2CAP Channels

The host or the device can initiate the setup of control and data L2CAP channels.

2.1.2 Bluetooth Packets

Bluetooth passes data over the air in the form of L2CAP packets, which should not be confused with NDIS or networking packets. The maximum length of packets supported by L2CAP must be the maximum MTU of the media + the maximum RNDIS header size. The device should fill in MaxTransferSize to the largest L2CAP message it can send, if the host has a smaller L2CAP maximum message size it should overwrite the returned information with its own maximum message size.

2.1.3 Bluetooth Flow Control

The flow control method used by Bluetooth Remote NDIS devices, is the flow control mechanism of Bluetooth Baseband protocol described in Part B Section 8.3 of Specification of the *Bluetooth System*.

2.2 Remote NDIS Device Support

Following is a description of minimal SDP record that will be needed for a Bluetooth Remote NDIS device.

2.2.1 Service Discovery Descriptor

The Remote NDIS device uses the standard Service Discovery descriptor as defined in Part E Specification of the *Bluetooth System*. The description for RNDIS is as follows: -

Item	Definition	Type/Size	Value	Attribute ID
ServiceClassIDList				0x0001
ServiceClass0		UUID/32-bit		
ProtocolDescriptorList				0x0004
Protocol0	L2CAP	UUID/32-bit	L2CAP	
Protocol1	PAN	UUID/32-bit	PAN	
ProtocolSpecficParameter0	Control Channel	Uint8	N = control channel #	
ProtocolSpecficParameter1	Data Channel	Uint8	N = data channel 1	
ProtocolSpecficParameterN	Data Channel	Uint8	N = data channel N	
ServiceName	Displayable text name	DataElement/String	String	

It is expected that PAN services can communicate with each other. It is possible for a Bluetooth device to have multiple PAN services. E.g. a cellphone could have a Wireless WAN server that gives Bluetooth devices access to the cellphone data network; in this case the ServiceName should be "WWAN", though a more descriptive name maybe used especially if multiple WWAN services are available to the cellphone. It can also have a PAN service that allows internal PAN services to communicate peer-to-peer between devices; in this case the ServiceName must be "PEER". A device must not advertise more than one PAN profile with a ServiceName of PEER.

2.2.2 L2Cap Channels

A Remote NDIS Bluetooth device will need to initiate or accept two or more L2CAP channels, a control channel and one or more data channels.

2.2.3 Messages

Messages to the device will be sent in the form of L2CAP PDUs. The device will be sent a message on the control channel from the host. The device then sends the host on the control channel the response.

A typical transaction for a Bluetooth RNDIS connectionless device would be: -

1. Host issues a Bluetooth WRITE on the control channel, with the payload of *NdisQueryRequest*. The RNDIS_OID value in the NdisQueryRequest is set to OID_GEN_MEDIA_CONNECT_STATUS.
2. Device receives the Bluetooth data, decodes the *NdisQueryRequest* and does the necessary actions to determine the connection status. When the device has the information, it issues a Bluetooth WRITE on the control channel consisting of an NdisQueryResponse (in this case the OID_GEN_MEDIA_CONNECTION_STATUS).

2.3 Bluetooth Peer-to-Peer RNDIS

Since Bluetooth is a peer-to-peer system and the SDP record does not define a difference between the host and device system, the Bluetooth microport should be capable of working against itself. A host only RNDIS microport only need to initiate certain messages and will only receive certain messages. As a microport that can be a host, a device or both, it needs to process all messages that can be received by either a host or a device. It is necessary for the microport to deal with these messages. It is expected that the Bluetooth microport will act as only a host microport when connected to a cellphone and as a dual host/device when connected to another machine that is also running the microport. Care is required in the design of the microport so oscillations in processing messages do not occur.
Remote NDIS defines the format for the REMOTE_NDIS_PACKET message including space to transport NDIS OOB and per packet information fields. The per packet information fields are only supplied by NDIS when the remote driver specifies it supports the functionality. The OOB information is only supported for particular media types. For Ethernet peer-to-peer emulation neither OOB or per packet information is required. In this case 44 bytes for offsets and lengths of packet information fields are not required. This means that for a 1514 byte Ethernet MTU L2CAP is required to have a minimum of 1554 byte MTU and wastes approximately 3% of the Bluetooth bandwidth. This could be an issue for slow Bluetooth links. This can be optimized if DataOffset is 4 assume that the rest of the RNDIS_PACKET header is NULL. This reduces the data overhead to 16 bytes or 1%.
In operation, it is expected that the Bluetooth microport be loaded when a remote Bluetooth support PAN service is in range and can be communicated to. Loading of the microport will cause an initialization message to be sent from the microport. When the remote device is a cellphone it will response with an initialization complete message. When the remote device is another Windows machine this machine will also generate an initialization message.
This section does not deal with the normal RNDIS processing but only with the additional messages received when acting as a device or as both a host and a device. The extra messages the microport can receive are: -

REMOTE_NDIS_INITIALIZE_MSG
REMOTE_NDIS_QUERY_MSG
REMOTE_NDIS_SET_MSG
REMOTE_NDIS_RESET_MSG
REMOTE_NDIS_KEEPALIVE_MSG

2.3.1 REMOTE_NDIS_INITIALIZE_MSG

The microport should generate an REMOTE_NDIS_INITIALIZE_CMPLT. The microport should completely initialize itself either on sending the initialize message or on receiving the initialize complete message. It should not do any initialization on receiving a REMOTE_NDIS_INITIALIZE_MSG.

2.3.2 REMOTE_NDIS_QUERY_MSG

The microport should respond depending on the OID in the query. Any NDIS OID may be supported if required but the following OIDs should be supported at least locally by the microport: -

2.3.2.1 OID_GEN_SUPPORTED_LIST

Return a list of RNDIS mandatory OIDs plus any optional OIDs supported by the microport. The remote device is allowed to return not supported when the microport returns the list of OIDs it supports.

2.3.2.2 OID_GEN_HARDWARE_STATUS

Return the status of the microport, one of the following codes: -

NdisHardwareStatusReady
NdisHardwareStatusInitializing
NdisHardwareStatusReset
NdisHardwareStatusClosing
NdisHardwareStatusNotReady

Optional, the microport must return **NdisHardwareStatusReady** except when sending or receiving a REMOTE_NDIS_RESET_MSG when it must return **NdisHardwareStatusReset** if the remote device doesn't support this OID.

2.3.2.3 OID_GEN_MEDIA_SUPPORTED

The microport must support **NdisMedium802_3**; it may optionally support other media. Optional, the microport should return **NdisMedium802_3** if the remote device doesn't support this OID.

2.3.2.4 OID_GEN_MEDIA_IN_USE

Return the current media in use, which must include **NdisMedium802_3**, optional.

2.3.2.5 OID_GEN_MAXIMUM_FRAME_SIZE

Return Ethernet maximum frame size for NdisMedium802_3. The microport must return the MTU from L2CAP if the remote device does not support this OID.

2.3.2.6 OID_GEN_LINK_SPEED

Return speed depending on the number of slots that can be used by the L2CAP channel. This requires an API to L2CAP to obtain this information. Optional because the microport must get the link speed from the local L2CAP channel.

2.3.2.7 OID_GEN_TRANSMIT_BLOCK_SIZE

Return Ethernet maximum frame size for NdisMedium802_3, optional.

2.3.2.8 OID_GEN_RECEIVE_BLOCK_SIZE
Return Ethernet maximum frame size for NdisMedium802_3, optional.

2.3.2.9 OID_GEN_VENDOR_ID
Return the microport vendor IEEE code or the vendor code from the Bluetooth radio MAC address or 0xFFFFFF, optional.

2.3.2.10 OID_GEN_VENDOR_DESCRIPTION
Return a vendor description string. E.g. "Microsoft Corp. RNDIS Bluetooth Microport", optional.

2.3.2.11 OID_GEN_CURRENT_PACKET_FILTER
Return list saved in variable that is set from the REMOTE_NDIS_SET_MSG message. Optional, the microport should return the set value.

2.3.2.12 OID_GEN_MAXIMUM_TOTAL_SIZE
Return maximum Ethernet packet size for NdisMedium802_3, optional.

2.3.2.13 OID_GEN_MAC_OPTIONS
Return 0, optional. The microport must return 0 if the remote device doesn't support this OID.

2.3.2.14 OID_GEN_MEDIA_CONNECT_STATUS
Since the microport is not loaded when the remote device is not in range, nor is the microport loaded when the interface is disabled, this OID can always return **NdisMediaStateConnected**. Optional, the microport should return **NdisMediaStateConnected** if the remote device doesn't support this OID.

2.3.2.15 OID_GEN_XMIT_OK
Microport needs to keep a count of number of successfully transmitted packets and return the value here, optional.

2.3.2.16 OID_GEN_RCV_OK
Microport needs to keep a count of number of successfully received packets and return the value here, optional.

2.3.2.17 OID_GEN_XMIT_ERROR
Microport needs to keep a count of number of failed transmitted packets and return the value here, optional.

2.3.2.18 OID_GEN_RCV_ERROR
Microport needs to keep a count of number of failed received packets and return the value here, optional.

2.3.2.19 OID_GEN_RCV_NO_BUFFER
Microport needs to keep a count of number of packets that failed to be received due to lack of buffers and return the value here, optional.

2.3.2.20 OID_802_3_PERMANENT_ADDRESS
Return the MAC address of the Bluetooth radio. The microport must obtain from L2CAP the MAC address of the remote Bluetooth device if the remote device does not support this OID.

2.3.2.21 OID_802_3_CURRENT_ADDRESS

Return the MAC address of the Bluetooth radio. There is no requirement for a configured option to change the MAC address. The microport must obtain from L2CAP the MAC address of the remote Bluetooth device if the remote device does not support this OID.

2.3.2.22 OID_802_3_MULTICAST_LIST

Return the list of addresses saved by the set command. Note, it is optional for the microport to save any multicast filters and may rely on the receiver device filtering. The microport should return an empty address filter list if the remote device does not support this OID.

2.3.2.23 OID_802_3_MAXIMUM_LIST_SIZE

Return the number of multicast filters supported, optional. The microport should return 0 if the remote device does not support this OID.

2.3.2.24 OID_802_3_RCV_ERROR_ALIGNMENT

Returns the number of receives errors due to alignment, optional.

2.3.2.25 OID_802_3_XMIT_ONE_COLLISIONS

Returns the number of receives errors due to a single collision, optional.

2.3.2.26 OID_802_3_MORE_COLLISIONS

Returns the number of receives errors due to multiple collisions, optional.

2.3.3 REMOTE_NDIS_SET_MSG

Any NDIS OID may be supported if required but the following OIDs should be supported at least locally by the microport: -

2.3.3.1 OID_802_3_MULTICAST_LIST

Return the list of multicast filters configured. Optional, the microport must return NDIS_STATUS_MULTICAST_FULL if the remote device doesn't support this OID.

2.3.3.2 OID_GEN_CURRENT_PACKET_FILTER

Save the values passed in. The variable should be zeroed on a reset. Optional, the microport can save this value locally if the remote device does not support this OID.

2.3.4 REMOTE_NDIS_RESET_MSG

The microport should make sure that the REMOTE_NDIS_RESET_MSG does not cause the system to send a REMOTE_NDIS_RESET_MSG. For Windows, the microport should respond with a REMOTE_NDIS_RESET_CMPLT message and do no work.

2.3.5 REMOTE_NDIS_KEEPALIVE_MSG

The microport should respond with a REMOTE_NDIS_KEEPALIVE_CMPLT message.